# OBSTACLE SENSING AUTONOMOUS MOBILE (OSAMO) ROBOT

EFFIDZ SYUFRIE BIN MOHD SHUKOR

ELECTRICAL & ELECTRONICS ENGINEERING
UNIVERSITI TEKNOLOGI PETRONAS

JUNE 2010

**Obstacle Sensing Autonomous Mobile (OSAMO) Robot**

By

Effidz Syufrie bin Muhd. Shukor

Dissertation Report submitted in partial fulfillment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

JUNE 2010

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

# CERTIFICATION OF APPROVAL

## OBSTACLE SENSING AUTONOMOUS MOBILE (OSAMO) ROBOT

By

Effidz Syufrie bin Muhd. Shukor

A project dissertation submitted to the

Electrical & Electronics Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by:

________________________

Dr. Noohul Basheer Zain Ali

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2010

# CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

EFFIDZ SYUFRIE BIN MUHAMAD SHUKOR

# ABSTRACT

The objective of this project is to design and produce an Obstacle Sensing Autonomous Mobile (OSAMO) Robot that can be use in the industrial or for domestic purpose. Robots may be used to perform tasks that are too dangerous or difficult for human, such as radioactive waste clean-up, or maybe perform a simple task and various daily activities domestically such as delivers some package to the specific destination or cleaning the house. The Scope of Study will cover areas of research done to fulfil design requirements of the project with its objective and functionality. The areas of design requirements will include movement mechanism, speech and voice recognition, sensors, controller and programming. Consultation with supervisors and other FYP students had aid the author in completing the project.

## ACKNOWLEDGEMENTS

First and foremost the author would like to take this opportunity to express her appreciation to all the parties that is involved in making Final Year Project (FYP) a success. The undergoing of this project would have not been possible without the assistance and guidance of certain individuals and organization whose contributions have helped in its completion.

The author would like to express his deepest appreciation to his project supervisor, Dr. Noohul Basheer Zain Ali for his endless support, guidance and consultation regarding this project throughout the year.

Special thanks to lab technologist; Mdm. Siti Hawa who has given countless technical support to the author in providing both advice and components, hardware, or software the author requires in completion of the project.

Last but not least, the author would like to give out her thanks to all those who have contributed directly or indirectly to the success of this project.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Background of Study

A robot can be defined as an electro-mechanical machine that can be programmed to perform some specific tasks autonomously. [1] A robot doesn't need to possess intelligence, but does need a basic set of instructions that enable it to function in the environment it is placed in. Robots may be used to perform tasks that are too dangerous or difficult for human, such as radioactive waste clean-up, or maybe perform a simple task and various daily activities domestically such as delivers some package to the specific destination or cleaning the house. Robots also can be used to perform repetitive works that needed high precision and high focus as human tends to become careless and tired if works in a long time.

Robots today are being utilized in a wide variety of industrial applications. Any job that involves repetitiveness, accuracy, endurance, speed, and reliability can be done much better by robots, which is why many industrial jobs that used to be done by humans are increasingly being done by robots. For example, for the past 30 years or thereabouts robots have progressively taken over the fully automated production lines of the automobile industry, wherein a chassis of a vehicle is transported along a conveyor belt and is welded, affixed, painted, and assembled by a succession of robot stations. [2]

## 1.2 Problem Statement

On a daily basis, there are many activities needed to be done by human from a simple task of delivering package to its drop off locations; i.e. to deliver mails to the neighbors, or maybe a dangerous or difficult task to be completed in the volcano, and sometimes human just do not have the time to do it all. Either domestic or industrial use, the robot is the perfect candidate to replace human to perform this entire task thus can reduce the human labor. As the robot will work directly with human, a simple speech system will make the robot appears friendlier and funnier to the users.

However, the cost to build a robot is a major concern especially for domestic or small scale purpose. The success of designing an inexpensive autonomous robot would attract users domestically.

This has given the author an idea to design an inexpensive, autonomous robot for domestic use. The robot should be able to turn and move around, sensing obstacle, and also greet people in friendly manner.

## 1.3 Objective

The objective of this project is to design and construct a robot which would be able to turn and move around, able to avoid collision of obstacles along its pathway and also will greet people in a friendly manner. The robot also can be use to deliver packages to assign person or maybe to be a messenger to deliver a simple message to the assign person. Cost will be minimized to achieve minimal expenses throughout project.

On the first semester of the FYP, the author will make a lot of research in literature review in designing a wholly functioning robot meets the required criteria, and on the second semester, the author will build the actual design and construction while modifications will be done along the way to improve the initial design. Programming and integrating systems will be done after the author has finished designing the whole robot.

There are a few objectives need to be achieved by the end of project completion.

### *1.3.1 To design a simple movement mechanism for the robot.*

The design of the movement mechanism for the robot must be simple yet effective. This is to make sure that the objective can be achieved in the given period. A simple mechanism also can reduce the complexity of the programming of the robot and reduce cost.

### *1.3.2 To be able to avoid obstacle collision.*

The robot moves by itself and will not be controlled by the human. So, the robot must be able to avoid obstacle collision along its pathway. Fail to do so will cause the damage to the robot and also failure to move due to the obstacle blocking its way thus the objective of the robot unmet.

## 1.4 Scope of Study

The scope of study can be divided into four subsections. In which interdependency and coordination between all four subsections must be developed to produce a functioning mobile robot.

### *1.4.1 Controllers and Programming.*

Controllers can be assumed as the brain of the robots. The author will directly save the programming and all the instruction to the memory in the controller. The controllers will control the robot as per the programming and instructions that has been set by the author. The author will use PIC microcontroller rather than PLC microcontroller because the former is cheaper than the latter. The programming language required for this controller is C programming.

### *1.4.2 Electrical Circuits and Sensors*

Suitable sensors are needed for collision avoidance system. Electrical circuits are needed to supply regulated and stable power for the robot. The electrical circuit for the speech system is also needed to be done maybe by using speaker or buzzer.

### 1.4.3 Robotic Movement Mechanism

The movement mechanism will be provided by using proper motors and actuators. The motors and actuators are the "muscles" of the robot. A specific design of motors and actuators are needed to produce the desired output and tasks.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 History

The first recorded design of a humanoid robot was done by Leonardo Da Vinci (1945). Da Vinci's notebooks, rediscovered in the 1950s, contain detailed drawings of a mechanical knight now known as Leonardo's robot, able to sit up, wave its arms and move its head and jaw. [3]

The first known functioning robot was created in 1783 by Jacques de Vaucanson, who made an android that played the flute.

Nowadays, there are many robots that can do different tasks. The most recognizable is ASIMO (Advanced Step in Innovative Mobility) created by Honda, which is a humanoid robot that can walk, run and also can interact with human.

## 2.2 Overview

A robot is defined as an electro-mechanical machine that can be programmed to perform some specific tasks autonomously. A robot doesn't need to possess intelligence, but does need a basic set of instructions that enable it to function in the environment it is placed in. Robot can be use in the industrial or for commercial purpose.

Robots are used to perform tasks that are too dangerous or difficult for human, such as radioactive waste clean-up, or maybe perform a simple task and various daily activities domestically such as delivers some package to the specific destination or cleaning the house.

Robotic technology has improved vastly from a robot that can only do simple task to a robot that can operate the whole factory. Many factory jobs are now performed by robots. This has led to cheaper mass-produced goods, including automobiles and electronics industry which needed high precision workmanship.

## 2.3 Types of Robots

Generally, the types of robots can be divided into two categories in which:

1. Manipulator/Industrial robot: Robot stationed in one location and able to move other objects such as the robotic arm for material sorting process.
2. Mobile robot: Robots which are able to move about.

### *2.3.1 Manipulator/Industrial Robots*

A robot that is stationed in one location and able to move other objects such as the robotic arm for material sorting process for example factory automation with industrial robots for palletizing food products like bread and toast. These robots are able to perform quality of work due to fatigue or weariness and they are able to operate for long hours. Some examples of industrial robots used in the market include:

a) SCARA robot

Commonly used in assembly applications, this selectively compliant arm for robotic assembly is primarily cylindrical in design. It features two parallel joints that provide compliance in one selected plane. The name SCARA means Selective Compliance Assembly Robot Arm which was introduces in the late 1970 as a robot ideally suited for assembly task. One of the main reasons this robot excels is because of the compliance feature it offers. "Compliance" is a robotic term that means that the robot is capable of adjusting to accommodate misalignment. [4]

b) Spherical robot

It is able to handle machine tools, spot welding, die-casting, gas welding and arc welding. It is a robot whose axes form a polar coordinate system and create a spherical-shaped work envelope. [4]

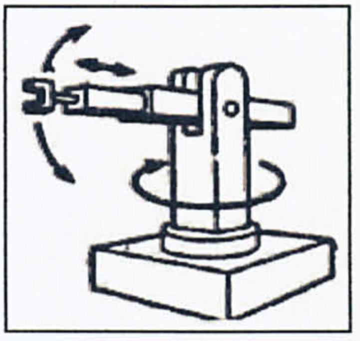


Figure 1: Diagram of spherical robot

c) Articulated robot

A robot which resembles a robotic arm, used for assembly operations, die-casting, fettling machines, gas welding, arc welding, and spray painting. It is a robot whose arm has at least three rotary joint. Each joint is called an axis and provides an additional degree of freedom, or range of motion. Industrial robots commonly have four or six axes. [4]

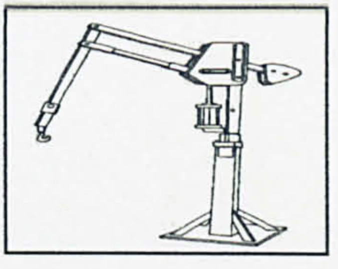


Figure 2: Diagram of articulated robot

d) Cylindrical robot

Normally used for assembly operations, handling at machine tools, spot welding and handling at die-casting machines. It is a robot whose axes form a cylindrical coordinate system. Cylindrical robots operate within a cylindrical-shaped work envelope. [4]

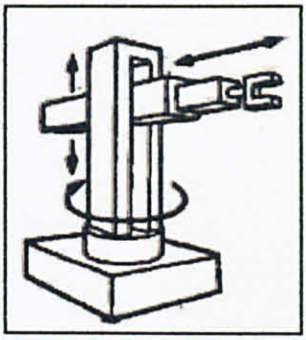


Figure 3: Diagram of cylindrical robot

### 2.3.2 Mobile Robots

Also known as Automated Guided Vehicles, or AGVs, these are used for transporting material over large sized places like hospitals, container ports, and warehouses, using wires or markers placed in the floor, or lasers, or vision, to sense the environment they operate in. These robots have the ability of performing tasks that are non-sequential and non-repetitive in environments that are complex. [2] Such ability allows the robots to be used in various fields such as:

#### a) Underwater exploration

Robotic underwater rovers are used to explore and gather information about many facets of our marine environment. Project Jeremy, is collaboration between NASA and Santa Clara University. An underwater telepresence remotely operated vehicle (TROV) was sent by scientist into the freezing Arctic Ocean waters to investigate the remains of a whaling fleet list in 1871. The TROV was cable operated, which carried power and instructions down to the robot and the robot returned video images. The TROV can also collect artifacts and gather information about the water conditions. [1]

b) Duct Cleaning

In the hazardous and tight spaces of a building's duct work, many hours can be spent cleaning relatively small areas if a manual brush is used. Robots have been used by many duct cleaners primarily in the industrial and institutional cleaning markets, as they allow the job to be done faster, without exposing workers to the harmful enzymes released by dust mites. For cleaning high-security institutions such as embassies and prisons, duct cleaning robots are vital, as they allow the job to be completed without compromising the security of the institution. Hospitals and other government buildings with hazardous and cancerogenic environments such as nuclear reactors legally must be cleaned using duct cleaning robots, in countries such as Canada, in an effort to improve workplace safety in duct cleaning. [1]

## 2.4 Building Blocks of the Robots

Robots and automation is made up of various building blocks, in which are interdependent to each other for the successful implementation and function of the robot. Failure in one of these blocks will affect the overall process of the automation. The building blocks of a robot consist of

1. Controllers
2. Sensors and Transducers
3. Analyzers
4. Actuators
5. Drivers

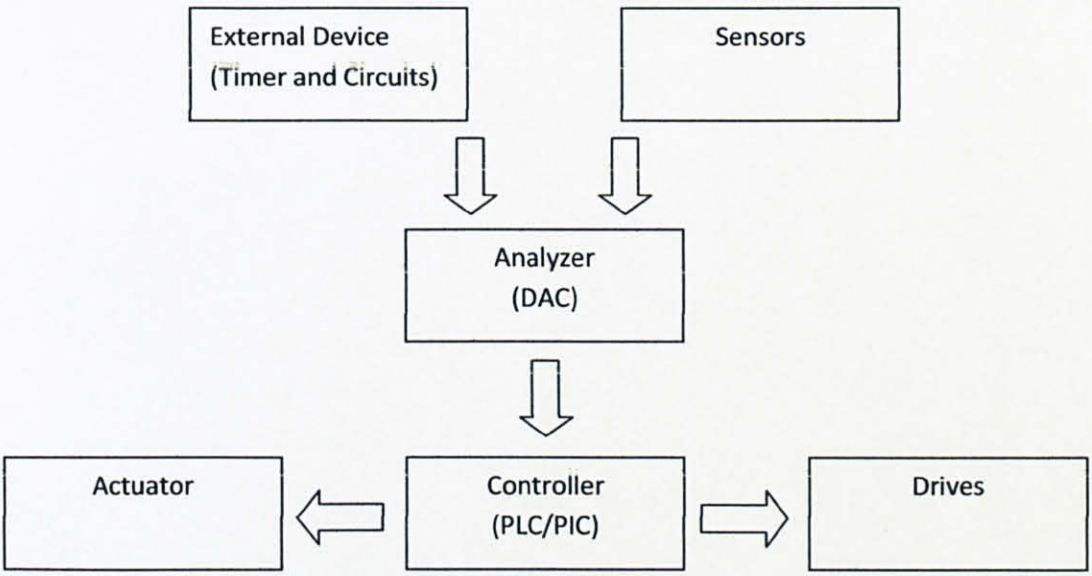



Figure 4: Building block components of robot and automation.

2.4.1 Controllers

Controllers can be assumed as the brain of the robots. The author will directly store the programming and all the instruction to the memory in the controller. The controllers will control the robot as per the programming and instructions that has been set by the author.

The author will use PIC 16F87X microcontroller rather than PLC microcontroller because the former is cheaper than the latter. The data and the information will be inputted to the controllers, processed according to the program defined and executed via the outputs. The programming language required for this controller is C programming.

Figure 5: PIC 16F87X controller

2.4.2 Sensors and Transducers

Sensors are the devices that generated output signal for the purpose of sensing a physical phenomenon. Manual switch will be used as the on-off switch for the robots. Another type of sensors is those that do not require physical contact. These proximity sensors are capable of sensing the presence of nearby objects without touching it; i.e. infrared sensors, photoelectric sensors and the ultrasonic sensors.

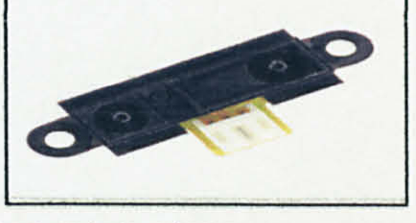


Figure 6: Example of sensors and transducers.

### 2.4.3 Analyzers

The function of the analyzers is to register and analyze the output signals produced by the sensors or devices. It will convert the input signal from analogue to digital by using ADC (analogue to digital converter) that can be understand by the robots.

### 2.4.4 Actuators

After the signal is sensed and analyzed, actuators will perform direct physical action for a particular process. Actuators may include solenoid and cylinders. Cylinders usually for pneumatic types and may be used in robots that grips. [5]

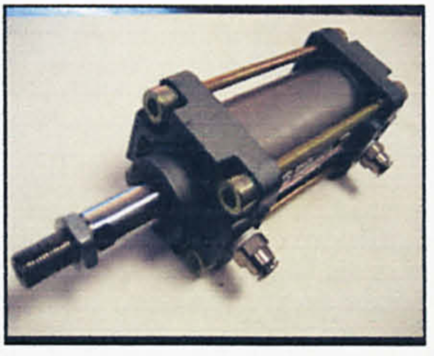


Figure 7: Example of actuators

### 2.4.5 Drives

Like actuators, drives take some action from the process command from the controller or to other analyzer. The difference is that actuators are used to affect a short, complete, discrete motion (usually linear) and drives executes more continuous movements typically motors. [5]

## 2.5 Controllers

Controller can be assumes as the brain for the robot. In order to build an effective but inexpensive robot, a suitable controller needed to be used. Below are the comparisons between the two microcontrollers that might be used in the author project.

### *2.5.1 Programmable Logic Controller (PLC)*

The PLC is a digitally operating electronic apparatus which uses a programmable memory for the internal storage of instructions by implementing specific functions such as logic sequencing, timing, counting, and arithmetic to control, through digital or analogue input/output modules. [6]

### *2.5.2 Programmable Input Controller (PIC)*

The PIC microchips are used for embedded designs. The Microcontroller is able to perform various functions according to the user programming codes (Assembly Language/C Programming). The microcontroller is able to perform task which is once difficult to be implemented on normal logic circuit.

Table 1: Comparison between PLC and PIC controllers.

| ***CRITERIA*** | ***PLC*** | ***PIC*** |
|---|---|---|
| **Control Device (Hardware)** | General purpose | Specific purpose but programmable |
| **Control Scale** | Medium and large | Small scale is likely to succeed |
| **Change or Addition of Specification** | Easy | Moderate (depends on the programming skills of the user) |
| **Delivery Period** | Almost immediate | Almost immediate |
| **Maintenance** | Easy | Moderate |
| **Reliability** | Very high | Not able to withstand industrial environment |
| **Economic Efficiency** | Advantage on large scale operation | Cheap |

## 2.6 Electronic Circuits

### *2.6.1 Collision Avoidance and Detection*

There are two different types of detection system. One is passive detection system, where the robot only changes direction after the collision happens. To make sure that the robot can avoid the obstacle, an active detection system must be used. The system would detect an obstacle from a defined distance while depending on the sensitivity of the sensors), and changes course of direction before collision.

#### *2.6.1.1 Infrared Sensor*

Light may always travel in a straight line but it bounces off nearly everything. This is advantageous to build an infrared collision detection system. The circuit below in Figure 8 is an example on how the infrared LED can be used for the purpose of detecting obstacle like a wall or an object in its path.

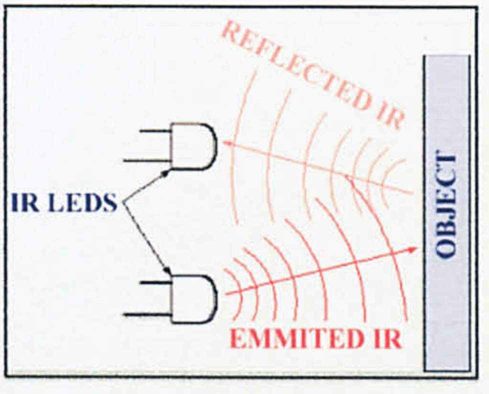



Figure 8: Using Infrared to Detect Obstacles

*a) Infrared Transmitter*

The transmitter LED is composed of an infrared LED (D2) in series with a 470 Ohm resistor, yielding a forward current of 7.5 mA. For this design, the light transmitting range can be varied from 1 to 10 cm depending on the ambient light in the environment. The resistor provides means to increase or decrease the range of the light transmission. [7]

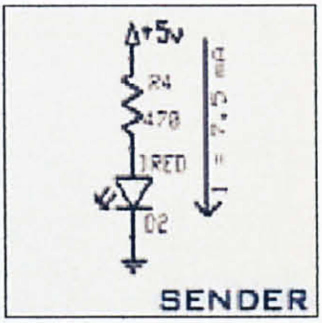



Figure 9: Circuit Diagram for Infrared Light Transmitter/Sender

*b) Infrared Receiver*

For the receiver part, the 2 resistor R5 and R6 forms a voltage divider which provides 2.5V at the anode of the infrared LED. When the LED (D1), receive the light from the reflection, the voltage drop increases, the cathode's voltage of D1 may go as low 1.4V depending on the light intensity. The voltage drop is detected by the OP-AMP. The variable resistor, R8 is adjusted so that the voltage at the positive input of the OP-AMP would be somewhere near 1.6V. So, when D1 receive the light, the voltage drop at the cathode of D1 become lower than 1.6V and the output becomes high. [7]

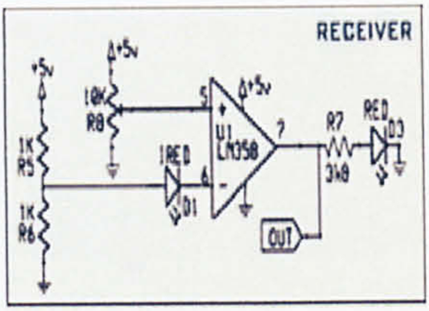



Figure 10: Circuit Diagram for Infrared Light Receiver

b) Ultrasonic Sensor

Sound can be used to detect the proximity of objects in much the same as for infrared light. Ultrasonic sound is transmitted from a transducer, is reflected by a nearby object, and then received by another transducer. The advantage of using sound is that it is not sensitive to objects of different color and light reflective properties. However, there are some materials that reflect sound better than others, and some even absorb sound completely. In comparison, proximity detection with sound is more fool proof.

The circuit below provides a practical circuit for building ultrasonic proximity detector. A stream of 40 kHz pulsed is produced by a 555 timer wired up as an astable multivibrator. The output of the 555 provides more than enough power for the transducer. A piece of foam between the transducers is needed to eliminate direct interference between the two. [8]

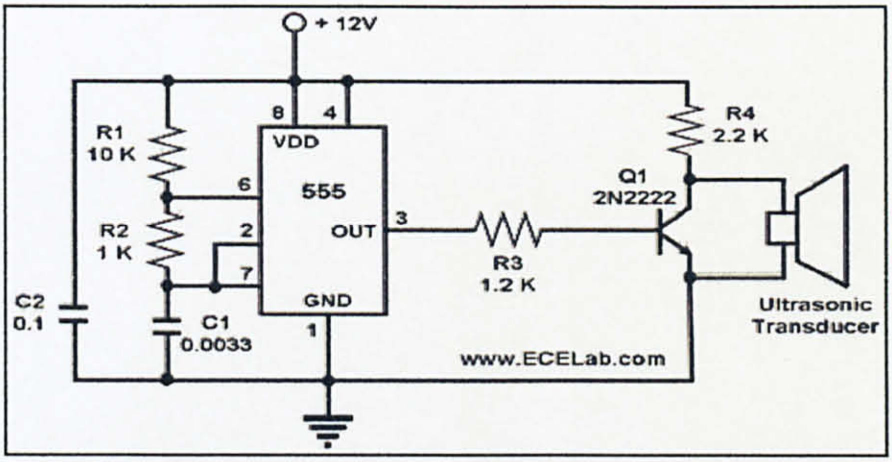



Figure 11: Schematic of 40 kHz ultrasonic transmitter.

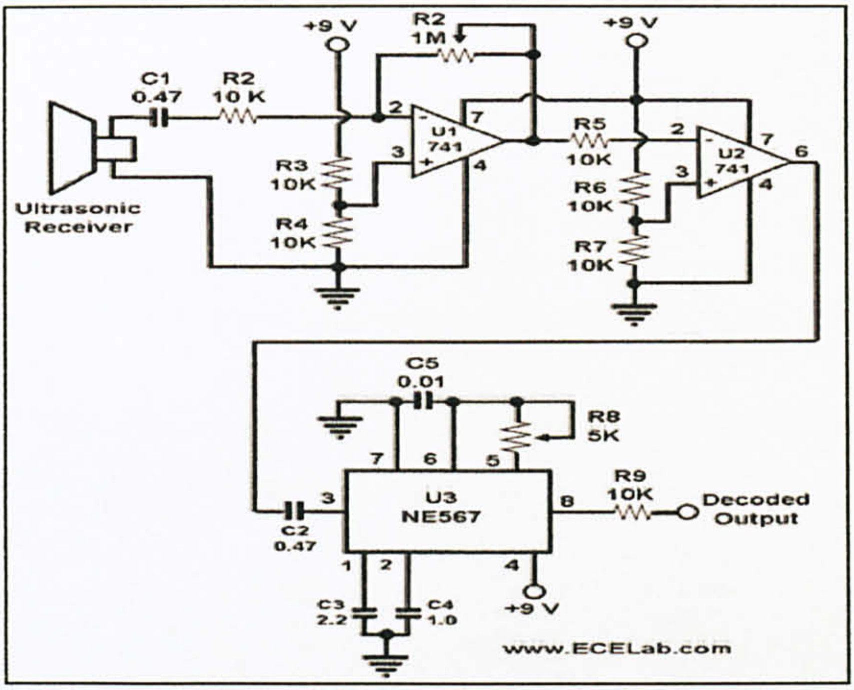



Figure 12: Schematic for an ultrasonic receiver and tone decoder.

## 2.7 Mechanical Drives

### *2.7.1 Choosing the Right Motor for the Job*

Motors are the muscle of the robot; whether to move other objects or to move the robot itself. There are several types of motor normally used by robots, which are the stepper motors, DC continuous motors and the servo motors.

*DC Motor:* The most common motor normally used to drive continuously in one direction. It only stops when the power supply is removed.

*Stepper Motor:* Application of power causes the shaft to rotate a few degrees and then stop. Continuous rotation of the shaft requires the power to be pulsed to the motor.

*Servo Motor:* Strong motors normally used as joints for robots. It can only move 180 degrees in direction.

### 2.7.2 Robot Locomotion with DC Motors

Most robot designs use two identical motor to spin two wheels. These wheels provide forward and backward momentum, as well as left and right steering. Stopping one motor, would allow the robot to change direction. Reversing both motors in relative to one another, robot turns by spinning on its wheel axis. This would cause sharp right and left turns. [9]

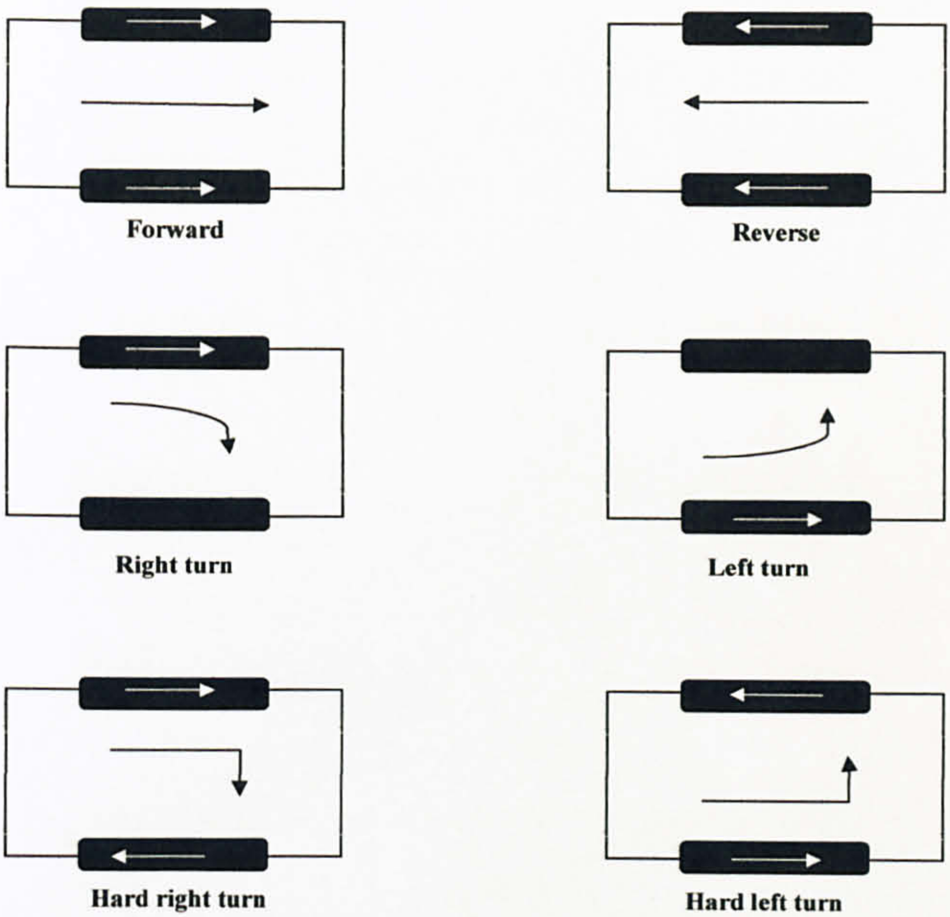



Figure 13: Direction of mobile robot with differential drive locomotion.

# CHAPTER 3

# METHODOLOGY

## 3.1 Procedure Identification

Below in Figure 14 is a flow chart describing the project methodology step-by-step in more detail, including intensive literature reviews, the mechanical and electrical design drawings and the structure fabrication for mobile robot.

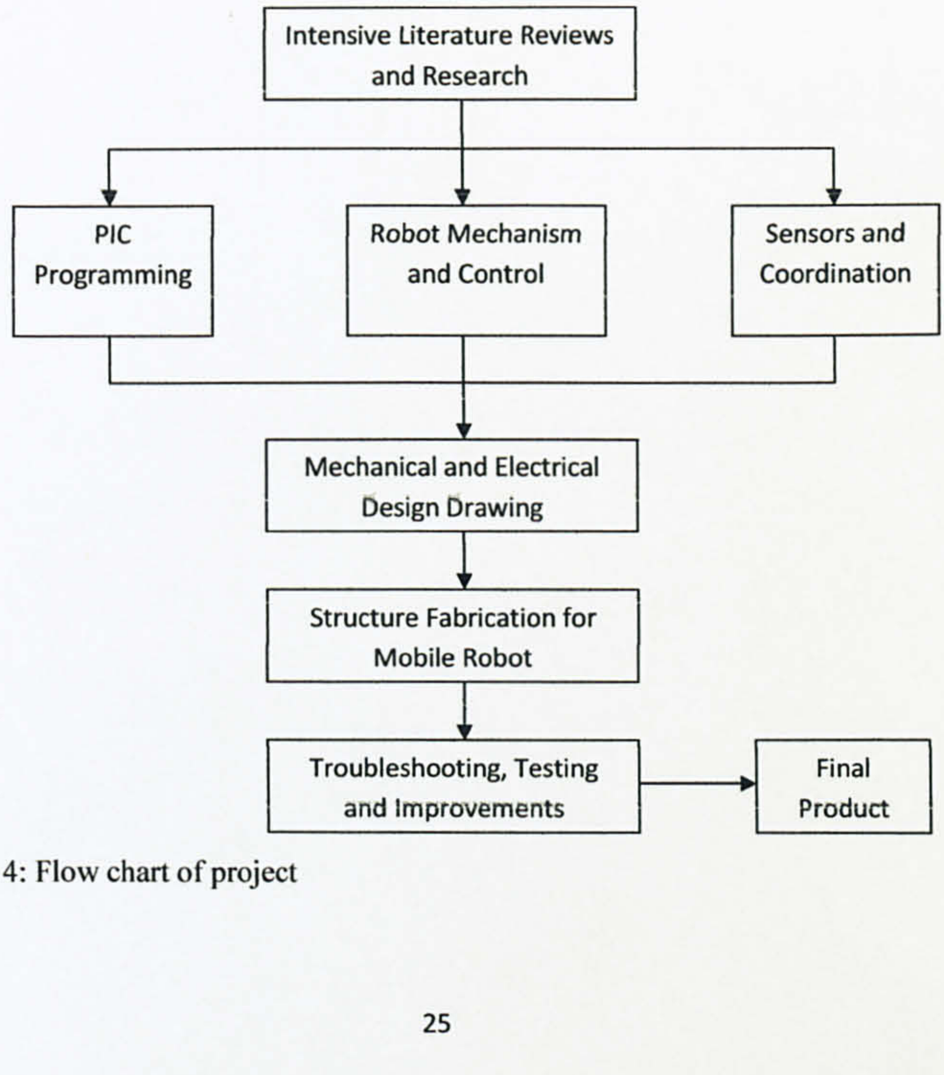



Figure 14: Flow chart of project

## 3.2 Tools and Equipments Required

Below in Table 1 are the expected tools and equipments needed for completion of the autonomous robot. These tools are obtainable for use from UTP Hardware Store and UTP Mechanical Engineering Laboratory and the software needed can be downloaded from the internet for free.

Table 2: Tools and Equipments Required

| No | Parts | Materials | Tools |
|---|---|---|---|
| 1. | Structure | 1. Metal | 1. Drill<br>2. Welding |
| 2. | Movement | 1. DC motor with gears(2)<br>2. Casters(2)<br>3. Nuts and Screws<br>4. Plastic wheel coupled(2) | 1. Screw drivers<br>2. Pliers |
| 3. | Power distribution | 1. Batteries(12V)<br>2. Connectors & Wires<br>3. Ribbon Cable | 1. Solder & flux<br>2. Multimeter<br>3. Screw drivers |
| 4. | Sensors | 1. Infrared LEDs<br>2. Ultrasonic Sensor<br>3. Electronic Components | 1. PSpice<br>2. Breadboard<br>3. Solder & flux<br>4. Multimeter<br>5. Oscilloscope |
| 5. | Microprocessor | 1. PIC Controller 16F87X<br>2. PIC Programmer Board<br>3. PIC 16F87X Target Board | 1. Breadboard<br>2. Electronics components<br>3. Oscilloscope<br>4. Programming software<br>5. Connectors |

## 3.3 Project Works

### *3.3.1 Intensive Literature Reviews*

Intensive literature reviews has been done on mobile robot fabrication. There are seven references that the author used for gaining knowledge on how to build a mobile robot. Resources from related books, internet and online journals have been accessed. The reviews are crucial to identify the method, tools and equipments that are needed for implementation of the robot. The literature reviews cover four important areas which are:

a) Robot Mechanism and Control

Mechanical trajectory and dynamics calculation needed to be done to identify the type of locomotion to be used, the torque of the motor needed and mechanisms that would be implemented to the design.

b) Sensors and Coordination

Identify the best suited sensor for the obstacle avoidance of the robot which is the active detection system.

c) PIC Programming Language

Learning the programming language (C++) which will be used to program the microcontroller selected for this robot which is the PIC chip.

### 3.3.2 Mechanical and Electrical Designs Drawings

The mechanical structures are drawn out by using specific software such as AutoCAD which includes the details in which the real robot would resemble. The electrical design will be done by using Multisim Electronic Workbench software for simulation purpose to test the theory and check the sufficiency of voltage and current supplied to the circuitry.

### 3.3.3 Structure Fabrication for Mobile Robot

The fabrication of this robot will be the final step after all the research has been done thoroughly. The body of the robot will be constructed and all the major components of the robot will be combines. After the fabrication completed, various tests will be done to make sure that the robot will function perfectly.

### 3.3.4 Troubleshooting and Testing

Troubleshooting is needed to ensure a functioning robot is produced at the end of the project period.

# CHAPTER 4

# RESULT AND DISCUSSION

## 4.1 The Overall Design

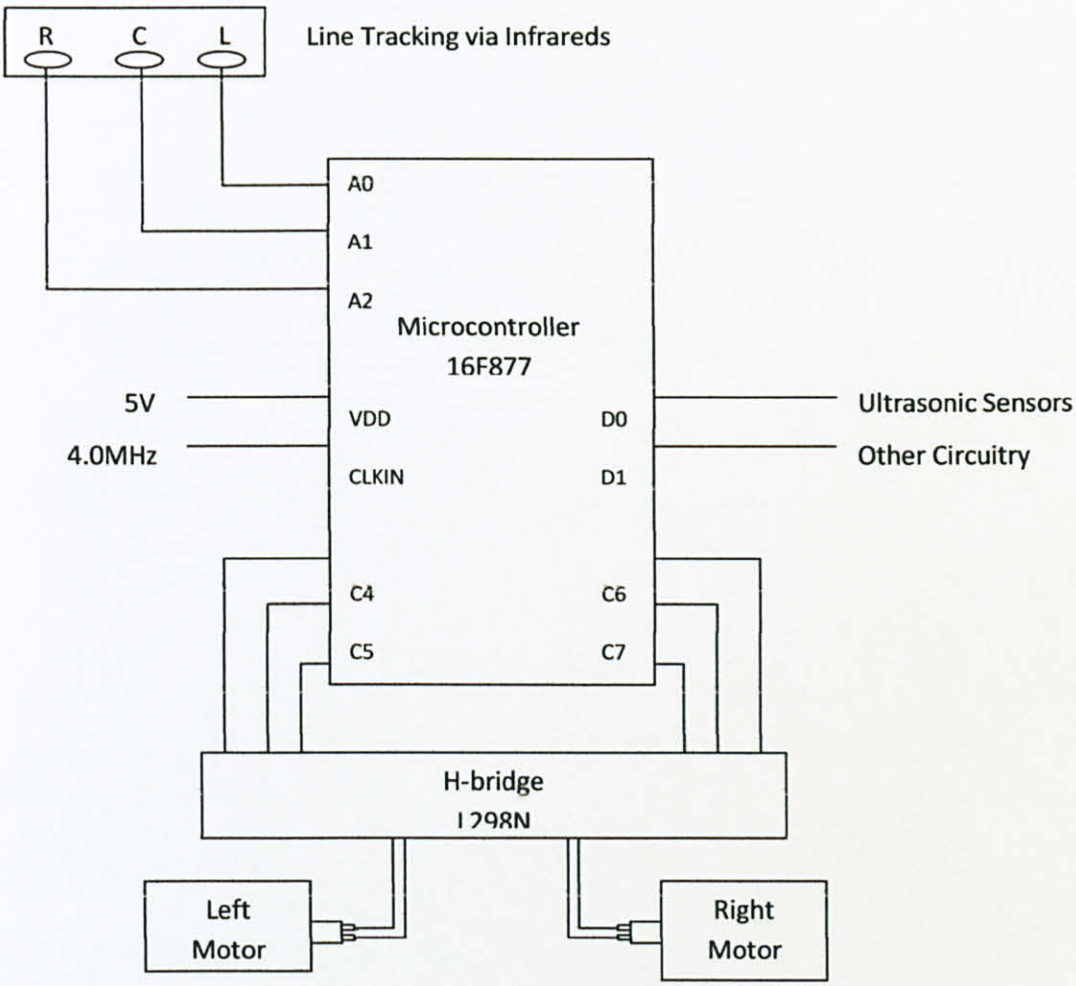



Figure 15: Overview of the robot circuitries with the microcontroller.

### 4.1.1 Component Functions and Interaction

With the reference to the Block Diagram shown in Figure 9, the functions of the components are described below:

a) *PIC 16F877 Microcontroller*

The overall process of the robot is determined by this chip. The chip receives inputs from the sensors and processes it to execute appropriate instructions according to the program written by the programmer. The controller also controls the drives such as the DC motor according to the path planning codes written.

b) *DC Motors*

The DC motors are used to move the robot about. The motor has sufficient amount of torque in order to successfully move the whole structure.

*c) Line Tracker*

The line tracker is essential for movement from point to point of the robot. The line tracker is design to follow white lines over work area to get around.

*d) Ultrasonic Receiver/ Transmitter*

The ultrasonic sensor is essential for the obstacle avoidance feature in this robot. When an obstacle is detected along the predetermined path of the robot, a signal would be sent to the microcontroller by the ultrasonic sensors. The obstacle avoidance program would then be executed to avoid any collision of the robot.

## 4.2 Structure and Design

The author has designed the robot to make sure that it meets the required criteria. On the front of the robot, there are Ultrasonic sensors (transmitter and receiver) to detect the obstacle. There is also a box on the robot to place the microcontroller board.

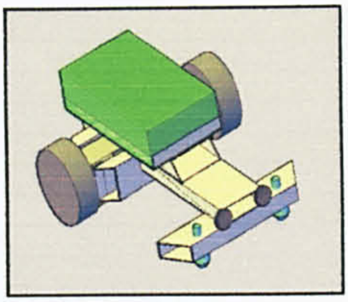


Figure 16: Design of the robot from the top view

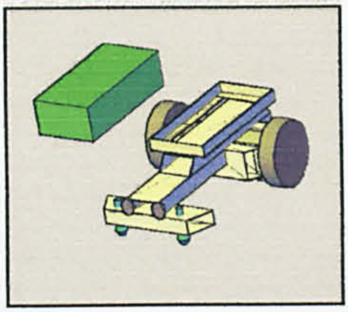


Figure 17: The box to place the microcontroller board

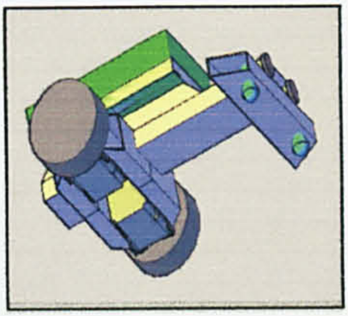


Figure 18: Design of the robot from the downside view

### 4.3 Microcontroller

The author had decided to make use of the 16F877 by the manufacturer Microchip. Please find attached in Appendix the datasheet for the microcontroller. The PIC16F877 shown in Figure 19, features 256 x 8 bytes of EEPROM data memory, self programming, an ICD, 5 channels of 10-bit Analog-to-Digital (A/D) converter, 2 additional timers, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and a Universal Asynchronous Receiver Transmitter (USART).

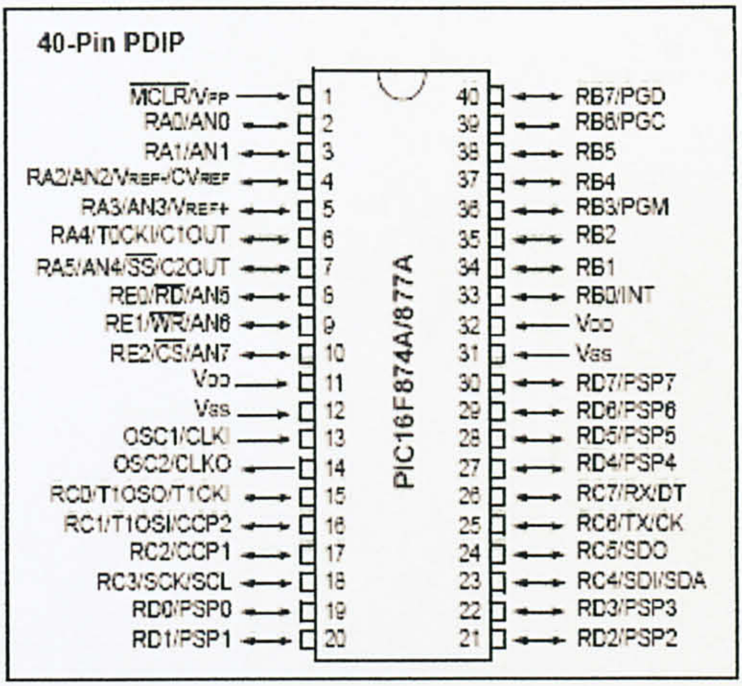



Figure 19: PIC 16F877 layout.

A programming board is required to transfer and load the program written in C language on a personal computer onto the chip. The Bumble Bee and PIC compiler was installed pm the computer and program written is transferred to the chip via Serial Cables. The program of the robot, load onto the microcontroller is attached in the Appendix.

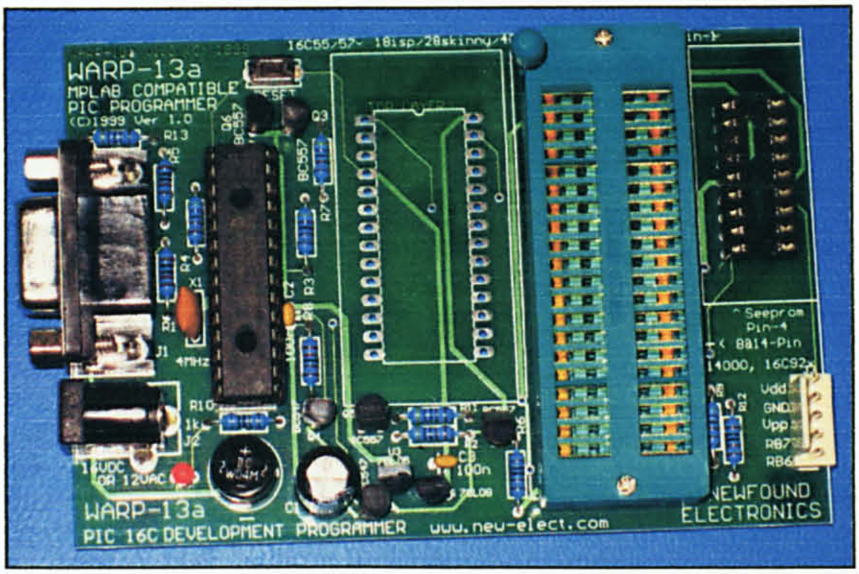



Figure 20: Warp 13 programmer board.

## 4.4 Line Tracking

### *4.4.1 Infrared Sensor*

The basic idea is to send infrared light through infrared LEDs (transmitter), which is then reflected by the embedded white lines on the floor. Then the reflected light is received by the exact same type Infrared LED (receiver).

The receiver LED produces a voltage difference across its leads when it is subjected to light. In order to detect a small voltage change, an Operational Amplifier (OP-AMP) is used.

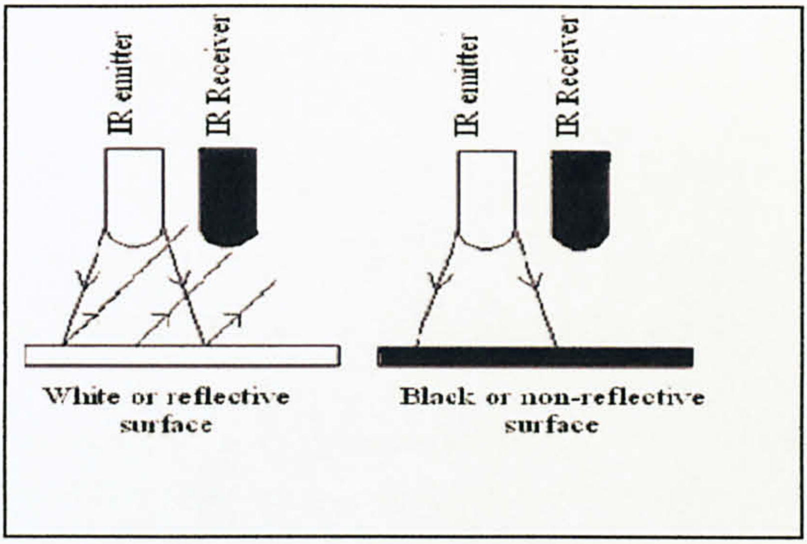



Figure 21: Line tracking using Infrared Light.

The line tracking circuitry will make use of one IR emitters on each side of the white line and one in the center of the line. These IR emitters are programmed to avoid the white line hence aligning the robots movement to actually follow the white line. This line tracking will act as guidance for the robot to move on the constructed testing track.

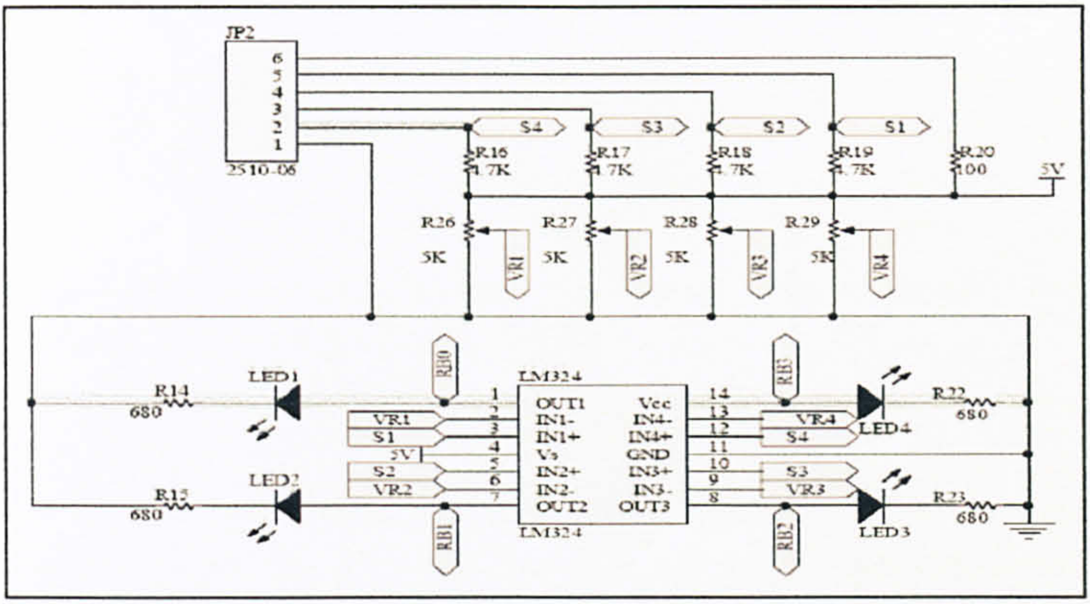



Figure 22: Line tracking circuitry

However sunlight can flood the sensors to the point of being useless. Our sun emits large quantities of light in the infrared spectrum, and although the human eye is blind to it, electronics are not. For obvious reasons the IR emitter/detector circuit is quite sensitive to even the smallest amount of sunlight. So, it is important to develop a shield that blocks the light.

## 4.5 Power Regulation

A power regulator circuit basically ensures control of the robots' power source. A power regulation circuit must do the following for the robot.

### *4.4.1 Regulate at a set voltage.*

For efficiency, optimally it would be best to use a power source closest (yet slightly above) the desired voltage input required. However this is rarely easy or even feasible. For a start, different electronics require different voltages. A microcontroller will require 5V, motor perhaps 12V, a voltage amplifier perhaps both 20V and -20V. Batteries are never at a constant voltage. A 6V battery will be at around 7V when fully charged, and can drop to 3 to 4V when drained.

### *4.4.2 Supply a minimum required amount of power.*

The sum required power of all the robot components needs to be below the amount the power circuit can supply. If power drops even for a fraction of a second below what the robot requires, things like the microcontroller could reset, or sensors would give bad readings, or motors won't work very well.

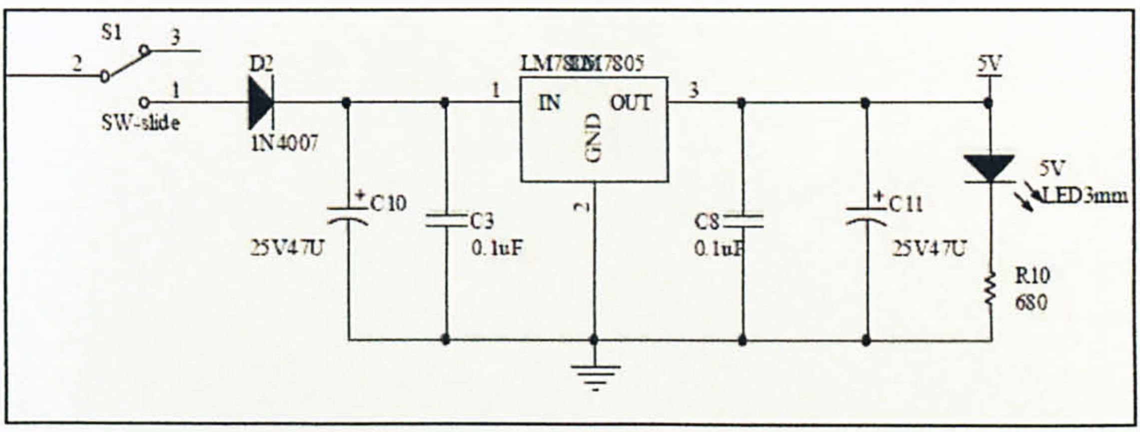



Figure 23: Power regulation schematic.

## 4.6 Collision Avoidance System

### *4.6.1 Ultrasonic Receiver*

The circuit has been designed based on the schematics shared on the internet by the electronic hobby kit webpage. The design circuit in Figure 24 allows the relay switch to activate when the receiver detects ultrasonic waves.

The circuit works based on the ultrasonic transducer when sensing ultrasonic signals. It converts it to electrical input with the same frequency. These signals are amplified by transistors T3 and T4. Then the amplified signals are the rectified and filtered. The filtered DC voltage is given to inverting pin of op-amp IC2. The non inverting pin is connected to a variable DC voltage in which the threshold value of ultrasonic signal received can be manipulated. The output of op-amp is used to bias transistor T5 which will then bias the transistor T6. When transistor T6 conducts, it actuates the relay. The relay can be used to control any electrical or electronic equipment such as the controller.

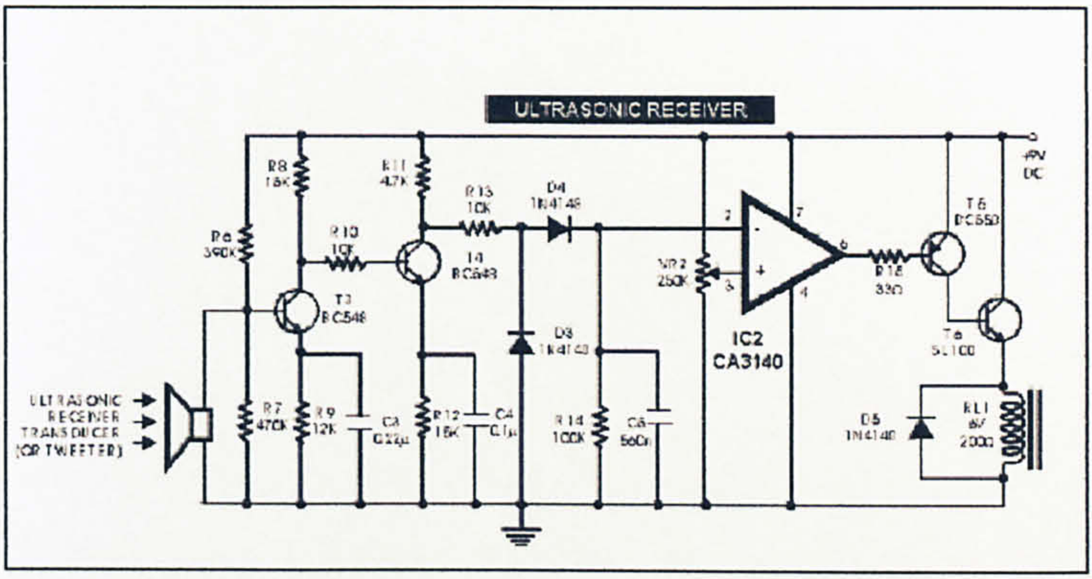



Figure 24: Schematic of ultrasonic receiver

### 4.6.2 Ultrasonic Transmitter

The ultrasonic transmitter uses a 555 based astable multivibrator. It oscillates at frequency of 40 to 50 kHz. The transmitter is powered from a 9V or 12V supply. The value of 4.7k with ±10k (pin 7) and the value of 18k (pin 6) and CI of 680picoFarad was chosen to generate a 40 kHz to 50 kHz frequency. The 40 kHz frequency can be calculated using the formula below.

$$f = \frac{1.44}{(2R2+R1)C1}$$

$$f = \frac{1.44}{(2X18k+15k)680pF}$$

$$f = 41.522 \text{ kHz}$$

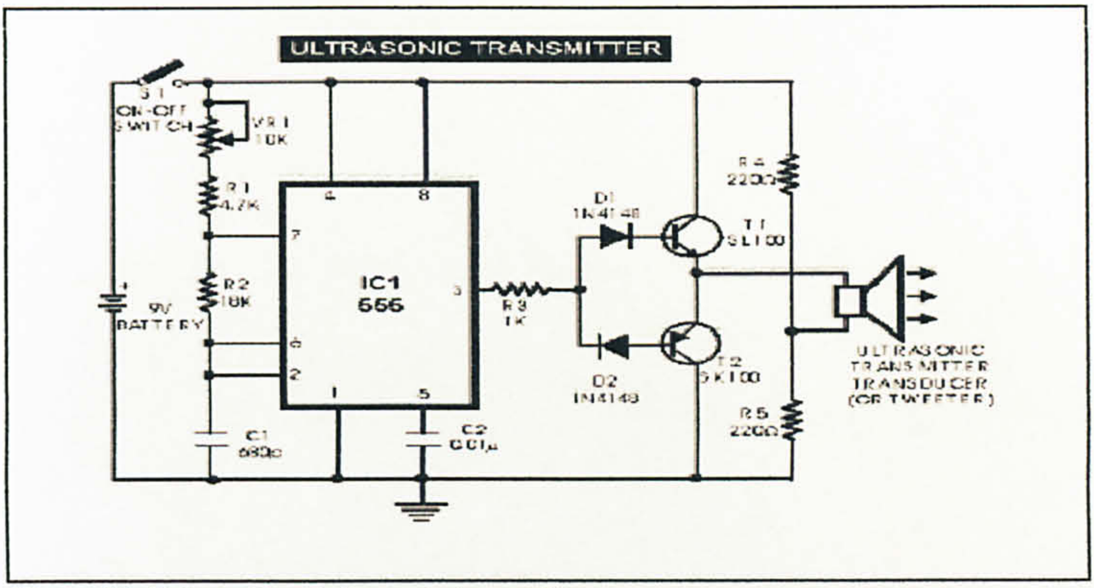



Figure 25: Schematic of ultrasonic transmitter.

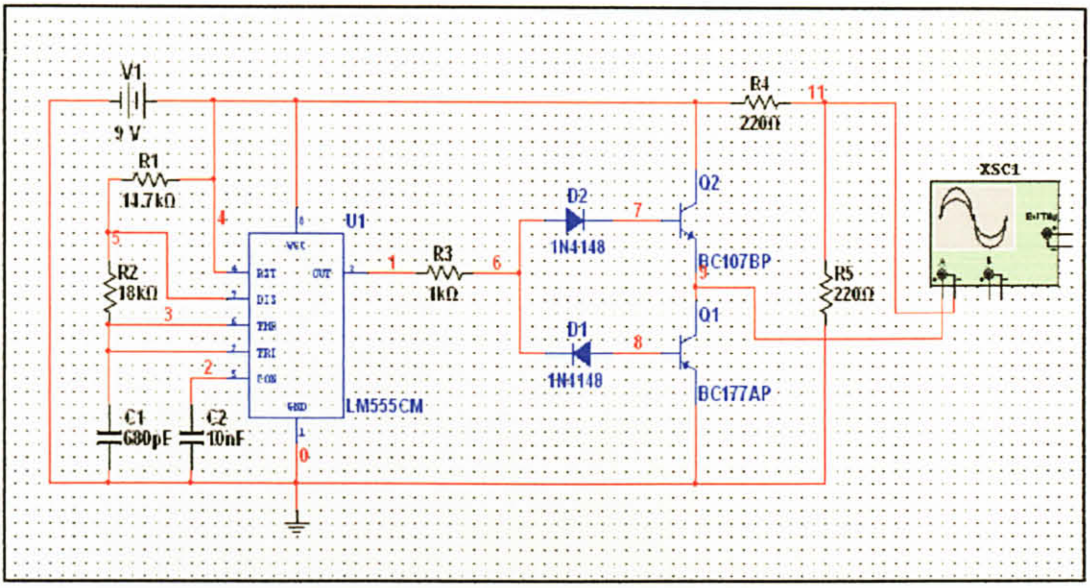



Figure 26: Simulation for Ultrasonic Transmitter on Electronic Workbench.

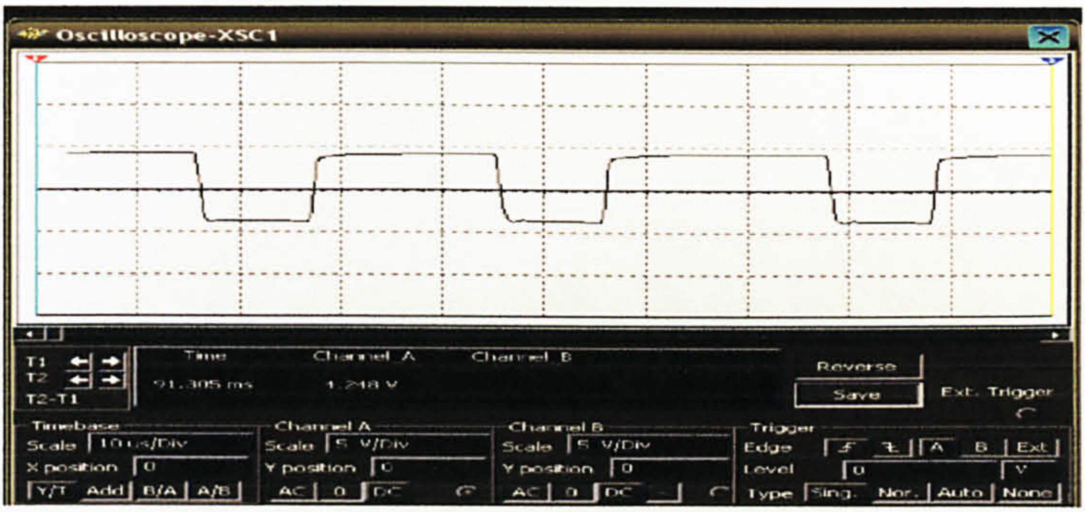



Figure 27: Simulation for 40 kHz frequency generation.

## 4.7 Source Code

The complete source code will be attached in Appendix A.

### *4.7.1 Choosing a mode*

```
        void main(void)
{
        unsigned char m=0,i =0;
        delay(20000);
        init();                                 // initiate cnfiguration and initial condition
        buzzer = 1;                             // inditcate the circuit is on with beep
        lcd_clr();                              // clear the LCD screen
        send_string("Select mode");             // display "select mode"
        lcd_goto(20);                           // move to 2nd line
        send_string(mode[m]);                   // display string according to the mode
        buzzer = 0;                             // stop beep
        while(1)                                // loop
        {
                if( !sw1)                       // if button SW1 is pressed
                {
                        while(!sw1);            // wait until button is released
                        m++;
                        if( m > 2) m = 0;               // if mode is added more than two, set to zero
                        lcd_goto(20);                   // start display at 20
                        send_string(mode[m]);           // display string depend on mode
                        send_string("      ");          // space to overwrite long words
                }
                if(!sw2)                                // if button SW2is pressed
                {
                        while(!sw2);                    // wait until button is released
                        switch(m)                       // check what is the current mode, execute the mode
                        {
                                case 0 :  line_follow();        // mode 1 : line follow
                                break;
                                case 1 :  object_follow();      // mode 2 : object following mode
                                break;
                                case 2 :  analog_sen();         // mode 3 : analog sensor mode
                                break;
                                default :  ;
                        }
                }
        }
}
```

Figure 28: Source code for Mode Selecting

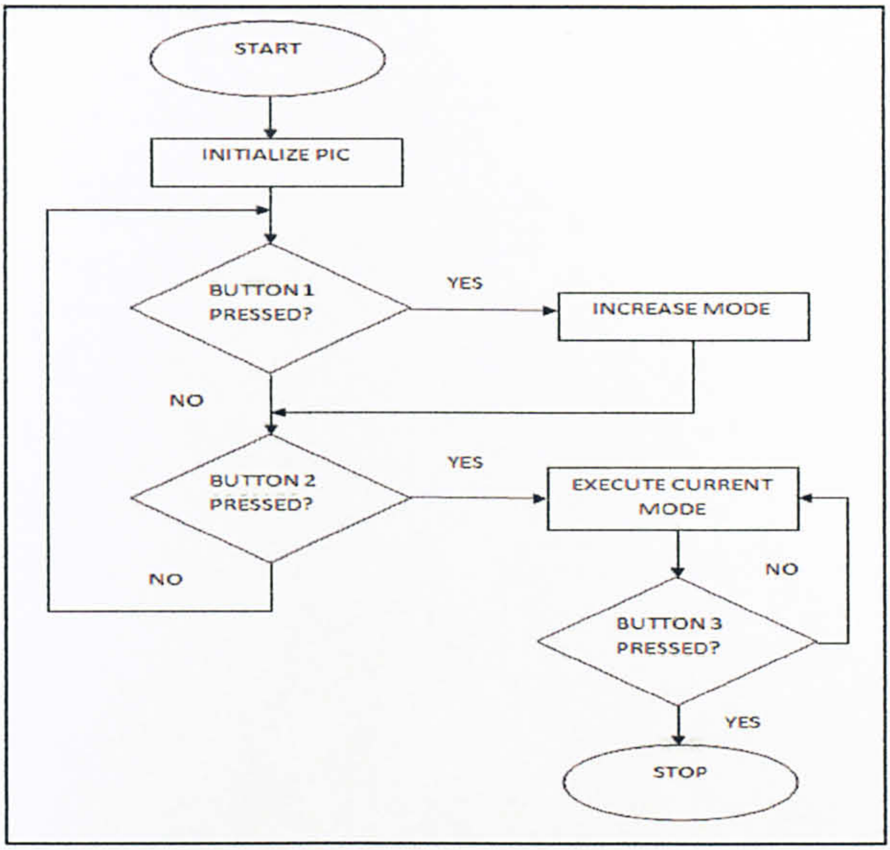



Figure 29: Mode selecting mechanism flow chart

Based on the flowchart in Figure 29 above, the steps in choosing the mode for the robot are as below:

1. Turn on the power for the microcontroller and the PIC will be initialized.
2. Press button 1/switch 1 to increase the mode. Starting mode is Mode 1. The current mode will be displayed on the robot's LCD.

   Mode 1: Line following

   Mode 2: Object following

   Mode 3: Analog Sensor/Obstacle Sensing
3. Press button 2/switch 2 to execute the mode.
4. Press button 3/switch 3 to reset the whole process.

### 4.7.2 Object Following Mode

```
void object_follow(void)
{
        int distance;                           // variable for distance measuring
        lcd_clr();                              // clear lcd
        send_string("Distance");                // display string

        while(1)
        {
                lcd_goto(20);                   // lcd goto 2nd line
                read_adc(CHANNEL1);             // read adc channel 1 ( analog distance sensor input)
                distance = result;              // assign distance as the result oft he reading
                dis_num(result);                // display the value


                if (distance> 333)              // check if distance more than 333
                {
                        stop();                 // stop
                        buzzer = 0;
                }
                else if (distance> 280)         // check if distance more than 280
                {
                        backward();             // backward with medium speed
                        SPEEDL = 255;
                        SPEEDR = 255;
                        buzzer = 0;
                }

        }
}
```

Figure 30: Source code for Object Following Mode

In Object Following mode, the robot will actually follow a particular object so that it can maintain the distance between the robot and the object. Backward in the source code does not mean that the robot will move in a reverse motion. The analog sensor is installed on the backside of the robot thus backward in this source code is actually a forward for the robot. From the source code, the robot will maintain its distance for about 333 IR Sensor values which is equal to 14.5cm from the object. The IR Sensor value can be converted into centimeters by using the equation 1 below,

$$Range = \left(\frac{3027.4}{IR\ Value}\right)^{1.2134} \dots\dots\dots\dots\dots \quad (1)$$

The robot will then follow the object if the object distances increased to 280 IR Sensor values which are equal to 18cm. The robot will stop moving when the distance with the object is back to 14.5cm.

### 4.7.3 Obstacle Sensing

```
void analog_sen(void)
{
    int distance;                          // variable for distance measuring
    lcd_clr();                             // clear lcd
    send_string("Distance");               // display string

    while(1)
    {
        lcd_goto(20);                      // lcd goto 2nd line
        read_adc(CHANNEL1);                // read adc channel 1 ( analog distance sensor input)
        distance = result;                 // assign distance as the result oft he reading
        dis_num(result);                   // display the value

        if (distance< 200)                 // check if distance less than 200
        {
            backward();                    // backward with full speed
            SPEEDL = 255;
            SPEEDR = 255;
            buzzer = 0;
        }
        else if (distance< 250)            // check if distance less than 250
        {
            backward();                    // backward with medium speed
            SPEEDL = 230;
            SPEEDR = 230;
            buzzer = 0;
        }
        else if( distance < 300)           // check if distance less than 300
        {
            stop();                        // stop
            buzzer = 0;
        }
        else                               // else, distance more than 300
        {
            right();                       //right with medium speed and on buzzer
            SPEEDL = 230;
            SPEEDR = 230;
            buzzer = 1;
        }
    }
}
```

Figure 31: Source code for Obstacle Sensing mode.

In obstacle sensing mode, when the robot sensed an obstacle in a distance of 200 IR Value (27cm), it will still move forward with full speed. When the distance from the obstacle becomes 250 IR Value (20.6cm), the robot speed will be decreased but still moving forward. Lastly, when the distance between the robot and the obstacle reach 300 IR Value (16.5cm), the robot will move to the right to avoid the obstacle collision.

# CHAPTER 5

# CONCLUSION

## 5.1 Conclusion

This project needs a very careful study and consistent works. There will be many obstacles that needed to be handled and overcame in completing the tasks. However with the guidance from the supervisor and other Electrical & Electronics Engineering Lecturers, it is hoped that the author can successfully completed the job according to the schedules.

In this project, an autonomous mobile robot is designed and constructed. The robot has achieved its main objectives which are being able to detect and avoid obstacles. This PIC controlled robot project will be the stepping-stone for the future UTP undergraduates to develop new and better robotics systems. Implementing knowledge gained from the class will be different from the knowledge gain from the hands-on experience. Having all the basics for developing a good robot in house will give strong future on robotics later.

## 5.2 Recommendations

Some of the recommendations the author plan to look into given no time constraint are as the following.

a) User Friendly Features

The author plans to add on user-friendly features to the robot, since its function to offer a helping hand to human daily activities. In order to make sure that the robot can communicate with human, a speech recognition component must be installed. A simple speech system will be installed that will make the robot can produce voices such as "excuse me", or "hello". The output devices that will be used are speaker and buzzer.

b) Neater Design

The author plans to install housing for the robot to not only protect the frame and circuitry, but also can result in a neater robot. The author also wants to give the robot a pleasant face which can make the robot become more attractive.

# REFERENCES


[1] Robot, 25 August 2009, Retrieve from http://en.wikipedia.org/wiki/Robot.

[2] Rita Putatunda, "*Types of Robots*", 9 Sept 2010
Retrieve from http://www.buzzle.com/articles/types-of-robots.html

[3] Mario Taddei, 2007. "*I Robot di Leonardo Da Vinci's Robots*", *1st Edition*, Leonardo3

[4] Robot Types, 9 Sept 2009, Retrieve from
http://www.robots.com/faq.php?question=robot+types

[5] Electraulics Actuator and Drives, Application Notes, 2009, REXA

[6] Programmable Logic Controller, 29 Sept 2009
Retrieve from http://en.wikipedia.org/wiki/Programmable_logic_controller

[7] Ibrahim Kamal, "Using an IR LED as a Sensor", 5 March 2010
Retrieve from http://www.ikalogic.com/ir_prox_sensors.php

[8] Ultrasonic Transmitter, 29 Sept 2009, Retrieve from
http://www.ecelab.com/circuit-ultrasonic-t.htm

[9] Angelles, Jorge 2006. "Fundamentals of Robotic Mechanical System"
Canada, John Wiley & Sons, Inc.

[10] Dominique Estival, 2009 "*Integration of Speech and Vision in a Small Mobile Robot*", University of Melbourne

[11] Gordon McComb, Myke Predko, 2006. *Robot Builder's Bonanza, 3rd Edition*, McGraw-Hill

[12] Bates, Martin 2004. "*Introduction to Microelectronic Systems, The PIC 16F84 Microcontroller*". Great Britain, Arnold.

[13] M. H. Ali, A. A. Shafie, M. F. Alias, 2009. "*Autonomous Soccer Robot Using Triple Infra-red Sensor for Ball Detection*", UIAM, Malaysia.

# APPENDICES

## APPENDIX A: Gantt Chart

| No. | Detail/Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Mechanical and Electrical Design Drawing | | | | | | | | | | | | | | | | | |
| 2 | Submission of Progress Report 1 | | | | | | | | | | | | | | | | | |
| 3 | Structure Fabrication for Mobile Robot | | | | | | | | | | | | | | | | | |
| 4 | Submission of Progress Report 2 | | | | | | | | | | | | | | | | | |
| 5 | Poster Exhibition (ElectrEx) | | | | | | | | | | | | | | | EXAM WEEK | | |
| 6 | Submission of Draft Report | | | | | | | | | | | | | | | | | |
| 7 | Submission of Dissertation (softbound) | | | | | | | | | | | | | | | | | |
| 8 | Troubleshooting and Testing | | | | | | | | | | | | | | | | | |
| 9 | Oral Presentation | | | | | | | | | | | | | | | | | |
| 10 | Submission of Project Dissertation (Hardbound) | | | | | | | | | | | | | | | | | |

Progress

Suggested milestone

# APPENDIX B

# PROGRAM SOURCE CODE

```
#include <pic.h>

__CONFIG ( 0x3F32 );


/*define*/


#define sw1            RE0
#define sw2            RE1
#define motor_ra       RC0
#define motor_rb       RC3
#define motor_la       RC4
#define motor_lb       RC5
#define s_left         RB0
#define s_mleft        RB1
#define s_mright       RB2
#define s_right        RB3
#define buzzer         RE2
#define rs             RB7
#define e              RB6
#define lcd_data       PORTD
#define b_light        RB5
#define SPEEDL         CCPR1L
#define SPEEDR         CCPR2L
#define CHANNEL1       0b10001001                      // AN1 ( Distance sensor )
#define RX_PIN         RC7
#define TX_PIN         RA2
#define BOT_ADD        100
```

```
/*global variable*/

unsigned char data[4] = {0};
const unsigned char line [] = {"1.LINE FOLLOW"};
const unsigned char OF[] = {"2.Object Following"};
const unsigned char AS[] = {"3.Analog Sensor"};
const unsigned char *mode [3] = {&line[0],&OF[0],&AS[0]};

unsigned int result;
unsigned int To=0,T=0,TH=0;
unsigned char REC;
unsigned char i=0,raw;

unsigned int us_value (unsigned char mode);

/*function prototype*/

void init(void);
void delay(unsigned long data);
void send_config(unsigned char data);
void send_char(unsigned char data);
void e_pulse(void);
void lcd_goto(unsigned char data);
void lcd_clr(void);
void send_string(const char *s);
void dis_num(unsigned long data);
```

```
void line_follow(void);
void object_follow(void);
void analog_sen(void);

void forward(void);
void stop (void);
void backward (void);
void reverse (void);
void left(void);
void right(void);

void read_adc(char config);
```

**/*interrupt prototype*/**

```
if (TMR0IF)
{
            TMR0IF = 0;
            To +=0x100;
    }
    if(RBIF)
    {
            RBIF = 0;
            if (RB4)
            {
                    TMR0 = 0;
                    To = 0;
```

```
                }
                else TH = TMR0 + To;
}
        if(RCIF)
        {
                RCIF = 0;
                if (RCREG == 'R') data[i=0]= RCREG;
                else if (RCREG == 100) data[i=0]= RCREG;
                if ((data[0] == 'R'))data [i++] = RCREG;
                if (i>4) i = 4;
        }
}


/*main function*/


void main(void)
{


        unsigned char m=0,i =0;
        delay(20000);
        init();
        buzzer = 1;
        lcd_clr();
        send_string("Select mode");
        lcd_goto(20);
        send_string(mode[m]);
        buzzer = 0;
```

```
    while(1)

    {

        if( !sw1)

        {

            while(!sw1);

            m++;

            if ( m > 2) m = 0;

            lcd_goto(20);

            send_string(mode[m]);

            send_string("      ");

        }


        if (!sw2)

        {

            while(!sw2);

            switch(m)

            {

                case 0 : line_follow();
                         break;

                case 1 : object_follow();
                         break;

                case 2 : analog_sen();
                         break;

                default :          ;

            }

        }

    }

}
```

```
/*initialize the microcontroller*/

void init()
{
        // ADC configuration
        ADCON1 = 0b10000100;
        // setup for capture pwm
        RBIE = 1;
        // motor PWM configuration
        PR2 = 255;
        T2CON =         0b00000100;
        CCP1CON =       0b00001100;
        CCP2CON =       0b00001100;

        // Tris configuration (input or output)
        TRISA = 0b00000011;
        TRISB = 0b00011111;
        TRISC = 0b10000000;
        TRISD = 0b00000000;
        TRISE = 0b00000011;

        // TMR 0 configiuation
        T0CS = 0;
        PSA = 0;
        PS2 = 1;
        PS1 = 1;
        PS0 = 1;
```

```
TMR0IE = 1;

TMR0 = 0;


//setup UART

SPBRG = 0x81;

BRGH = 1;

TXEN = 1;

TX9 = 0;

CREN = 1;

SPEN = 1;

RX9 = 0;

RCIE = 1;


// enable all unmasked interrupt

GIE = 1;

PEIE = 1;


// LCD configuration

send_config(0b00000001);

send_config(0b00000010);

send_config(0b00000110);

send_config(0b00001100);

send_config(0b00111000);


TX_PIN = 1;

b_light = 0;

buzzer = 0;
```

```
        stop();

        b_light = 0;

        buzzer = 0;

        stop();

}


/*mode 1: line follow*/


void line_follow()

{

        unsigned char memory;

        lcd_clr();

        send_string("Position");

        while(1)

        {

                if ((s_left==1)&&(s_mleft==0)&&(s_mright==0)&&(s_right==0))


                {       forward();
                                SPEEDL = 0;
                                        SPEEDR = 255;
                                                memory = PORTB&0b00001111;
                                                        lcd_goto(20);
                                                                send_string ("right  ");
                                                                }

                else if ((s_left==1)&&(s_mleft==1)&&(s_mright==0)&&(s_right==0))


                {       forward();
                                SPEEDL = 180;
                                        SPEEDR = 255;
                                                memory = PORTB&0b00001111;
                        lcd_goto(20);
                        send_string ("m_right2");
```

```
}
else if ((s_left==0)&&(s_mleft==1)&&(s_mright==0)&&(s_right==0))

{         forward();
                    SPEEDL = 200;
                              SPEEDR = 255;
                                        memory = PORTB&0b00001111;
          lcd_goto(20);
          send_string ("m_right1  ");
}

else if ((s_left==1)&&(s_mleft==1)&&(s_mright==1)&&(s_right==0))

{         forward();
                    SPEEDL = 200;
                              SPEEDR = 255;
                                        memory = PORTB&0b00001111;
          lcd_goto(20);
          send_string ("m_right1  ");
}

else if ((s_left==0)&&(s_mleft==1)&&(s_mright==1)&&(s_right==0))

{         forward();
                    SPEEDL = 255;
                              SPEEDR = 255;
                                        memory = PORTB&0b00001111;
          lcd_goto(20);
          send_string ("middle ");
}

else if ((s_left==0)&&(s_mleft==0)&&(s_mright==1)&&(s_right==0))

{         forward();
                    SPEEDL = 255;
                              SPEEDR = 200;
                                        memory = PORTB&0b00001111;
          lcd_goto(20);
          send_string ("m_left1   ");
}
```

```
else if ((s_left==0)&&(s_mleft==1)&&(s_mright==1)&&(s_right==1))

{       forward();
                SPEEDL = 255;
                        SPEEDR = 200;
                                memory = PORTB&0b00001111;
        lcd_goto(20);
        send_string ("m_left1   ");
}

else if ((s_left==0)&&(s_mleft==0)&&(s_mright==1)&&(s_right==1))

{       forward();
                SPEEDL = 255;
                        SPEEDR = 180;
                                memory = PORTB&0b00001111;
        lcd_goto(20);
        send_string ("m_left2 ");
}

else if ((s_left==0)&&(s_mleft==0)&&(s_mright==0)&&(s_right==1))

{       forward();
                SPEEDL = 255;
                        SPEEDR = 0;
                                memory = PORTB&0b00001111;
        lcd_goto(20);
        send_string ("left   ");
}

else if ((s_left==0)&&(s_mleft==0)&&(s_mright==0)&&(s_right==0))

{       forward();
                if ((memory == 0b00000001)||(memory ==
0b00000011)||(memory ==         0b0000010)||(memory == 0b0000111))

        {

                SPEEDL = 0;
                        SPEEDR = 255;
                        }
```

```
                    else if ((memory == 0b00001000)||(memory ==
                    0b0000100)||(memory == 0b00001100)||(memory == 0b0001110))

                    {

                              SPEEDL = 255;
                                        SPEEDR = 0;
                                        }

          }

          else if ((s_left==1)&&(s_mleft==1)&&(s_mright==1)&&(s_right==1))
                    {

                    forward();
                    }

     }

}
```

**/*mode 2: object following*/**

```
void object_follow(void)

{

     int distance;
          lcd_clr();
               send_string("Distance");


     while(1)

     {

          lcd_goto(20);
               read_adc(CHANNEL1);
                    distance = result;
                         dis_num(result);
                              if (distance> 333)
                                   {

               stop();
                    buzzer = 0;

          }
```

```
            else if (distance> 280)

            {

                    backward();
                            SPEEDL = 255;
                    SPEEDR = 255;
                    buzzer = 0;
            }
        }

}
```

**/*mode 3: analog sensor*/**

```
void analog_sen(void)

{

        int distance;

        lcd_clr();

        send_string("Distance");

        while(1)

        {

                lcd_goto(20);

                read_adc(CHANNEL1);

                distance = result;

                dis_num(result);

                if (distance< 200)

                {

                        backward();

                        SPEEDL = 255;

                        SPEEDR = 255;

                        buzzer = 0;
```

```
        }
        else if (distance< 250)
        {
                backward();
                SPEEDL = 230;
                SPEEDR = 230;
                buzzer = 0;
        }
        else if( distance < 300)
        {
                stop();
                buzzer = 0;
        }
        else
        {
                right();
                SPEEDL = 230;
                SPEEDR = 230;
                buzzer = 1;
        }
    }
}
```

**/*read adc*/**

```
void read_adc(char config)
{
	unsigned short i;
	unsigned long result_temp=0;
	ADCON0 = config;
	delay(10000);
	for(i=200;i>0;i-=1)
	{
		ADGO = 1;
		while(ADGO==1);
		result=ADRESH;
		result=result<<8;
		result=result|ADRESL;
		result_temp+=result;
	}
	result = result_temp/200;
	ADON = 0;
}
```

```
/*motor control functions*/

void forward ()
{
	motor_ra = 0;
	motor_rb = 1;
	motor_la = 0;
	motor_lb = 1;
}

void backward ()
{
	motor_ra = 1;
	motor_rb = 0;
	motor_la = 1;
	motor_lb = 0;
}

void left()
{
	motor_la = 1;
	motor_lb = 0;
	motor_ra = 0;
	motor_rb = 1;
}
```

```
void right()

{

        motor_la = 0;

        motor_lb = 1;

        motor_ra = 1;

        motor_rb = 0;

}


void stop()

{

        motor_la = 1;

        motor_lb = 1;

        motor_ra = 1;

        motor_rb = 1;

}
```

**/*LCD functions*/**

```
void delay(unsigned long data)

{

        for( ;data>0;data-=1);

}

void send_config(unsigned char data)

{

        rs=0;

        lcd_data=data;

        delay(400);
```

```
        e_pulse();
}
void send_char(unsigned char data)
{
        rs=1;
        lcd_data=data;

        delay(400);
        e_pulse();
}
void e_pulse(void)
{
        e=1;
        delay(300);
        e=0;
        delay(300);
}

void lcd_goto(unsigned char data)
{
        if(data<16)
        {
        send_config(0x80+data);
        }
        else
        {

                data=data-20;
                send_config(0xc0+data);
```

```
        }
}
                void lcd_clr(void)

{
        send_config(0x01);
        delay(350);
}
void send_string(const char *s)
{
        while (s && *s)send_char (*s++);
}
void dis_num(unsigned long data)
{
        unsigned char hundred_thousand;
        unsigned char ten_thousand;
        unsigned char thousand;
        unsigned char hundred;
        unsigned char tenth;
        hundred_thousand = data/100000;
        data = data % 100000;
        ten_thousand = data/10000;
        data = data % 10000;
        thousand = data / 1000;
        data = data % 1000;
        hundred = data / 100;
        data = data % 100;
        tenth = data / 10;
```

```
data = data % 10;


send_char(hundred_thousand + 0x30);
send_char(ten_thousand + 0x30);
send_char(thousand + 0x30);
send_char(hundred + 0x30);
send_char(tenth + 0x30);
send_char(data + 0x30);
}
```

# APPENDIX C

# PIC16F87X DATASHEET

# PIC16F87X

## 28/40-Pin 8-Bit CMOS FLASH Microcontrollers

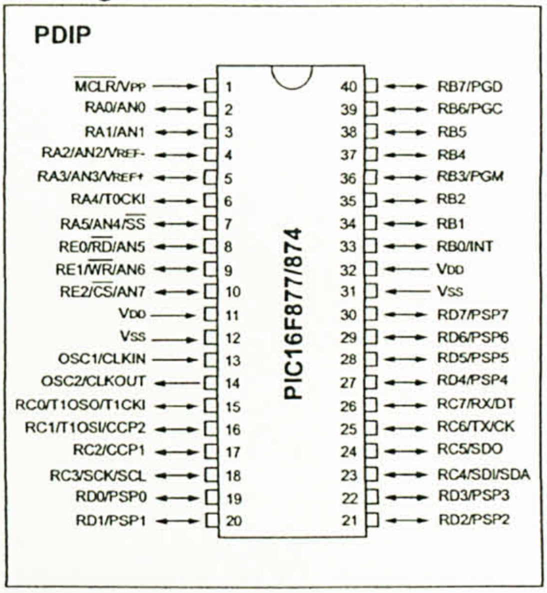
### Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

### Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

### Pin Diagram

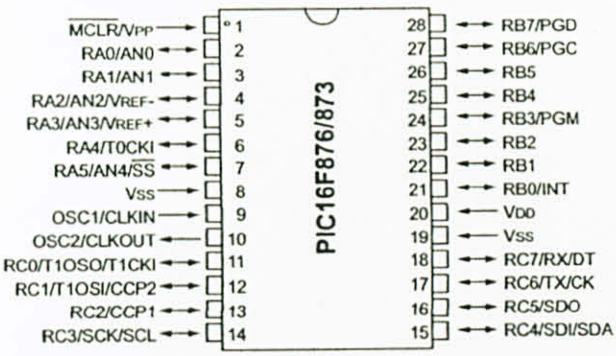



### Peripheral Features:

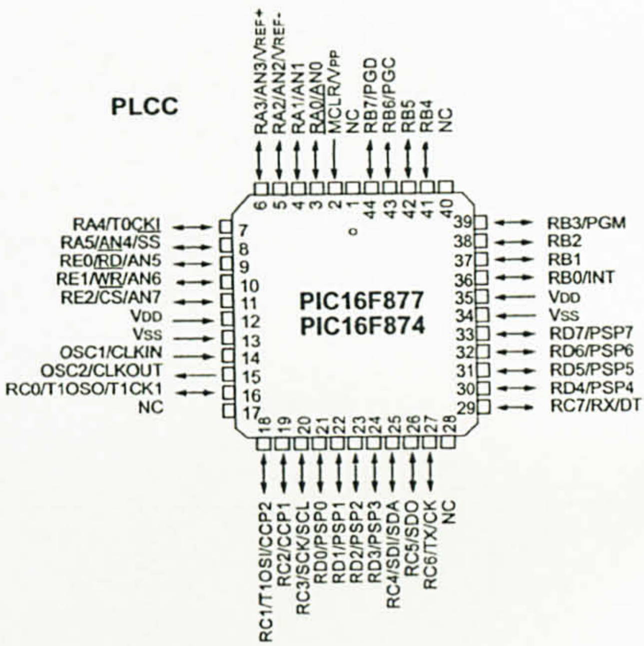
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I$^2$C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

## Pin Diagrams

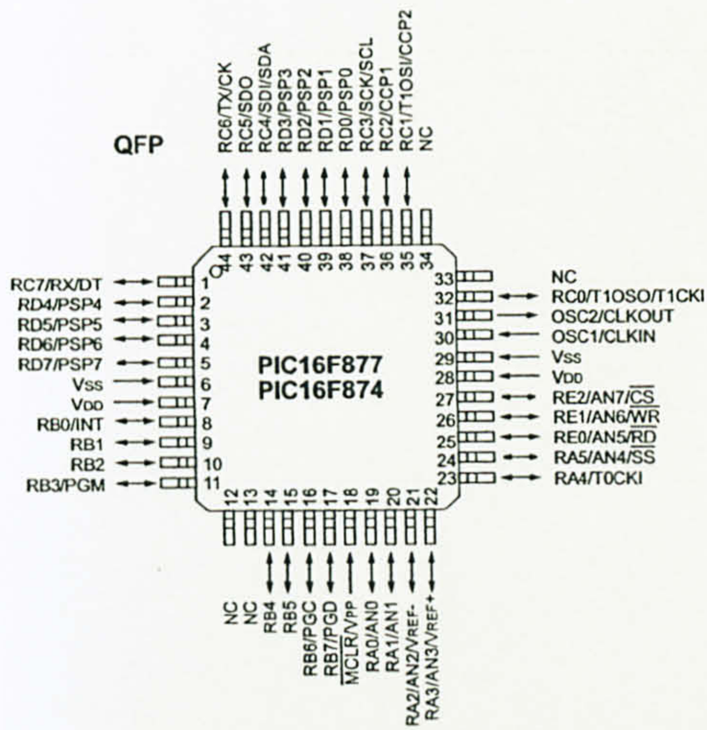
### PDIP, SOIC




### PLCC




### QFP



| Key Features PICmicro™ Mid-Range Reference Manual (DS33023) | PIC16F873 | PIC16F874 | PIC16F876 | PIC16F877 |
|---|---|---|---|---|
| Operating Frequency | DC - 20 MHz | DC - 20 MHz | DC - 20 MHz | DC - 20 MHz |
| RESETS (and Delays) | POR, BOR (PWRT, OST) | POR, BOR (PWRT, OST) | POR, BOR (PWRT, OST) | POR, BOR (PWRT, OST) |
| FLASH Program Memory (14-bit words) | 4K | 4K | 8K | 8K |
| Data Memory (bytes) | 192 | 192 | 368 | 368 |
| EEPROM Data Memory | 128 | 128 | 256 | 256 |
| Interrupts | 13 | 14 | 13 | 14 |
| I/O Ports | Ports A,B,C | Ports A,B,C,D,E | Ports A,B,C | Ports A,B,C,D,E |
| Timers | 3 | 3 | 3 | 3 |
| Capture/Compare/PWM Modules | 2 | 2 | 2 | 2 |
| Serial Communications | MSSP, USART | MSSP, USART | MSSP, USART | MSSP, USART |
| Parallel Communications | — | PSP | — | PSP |
| 10-bit Analog-to-Digital Module | 5 input channels | 8 input channels | 5 input channels | 8 input channels |
| Instruction Set | 35 instructions | 35 instructions | 35 instructions | 35 instructions |

## 1.0 DEVICE OVERVIEW

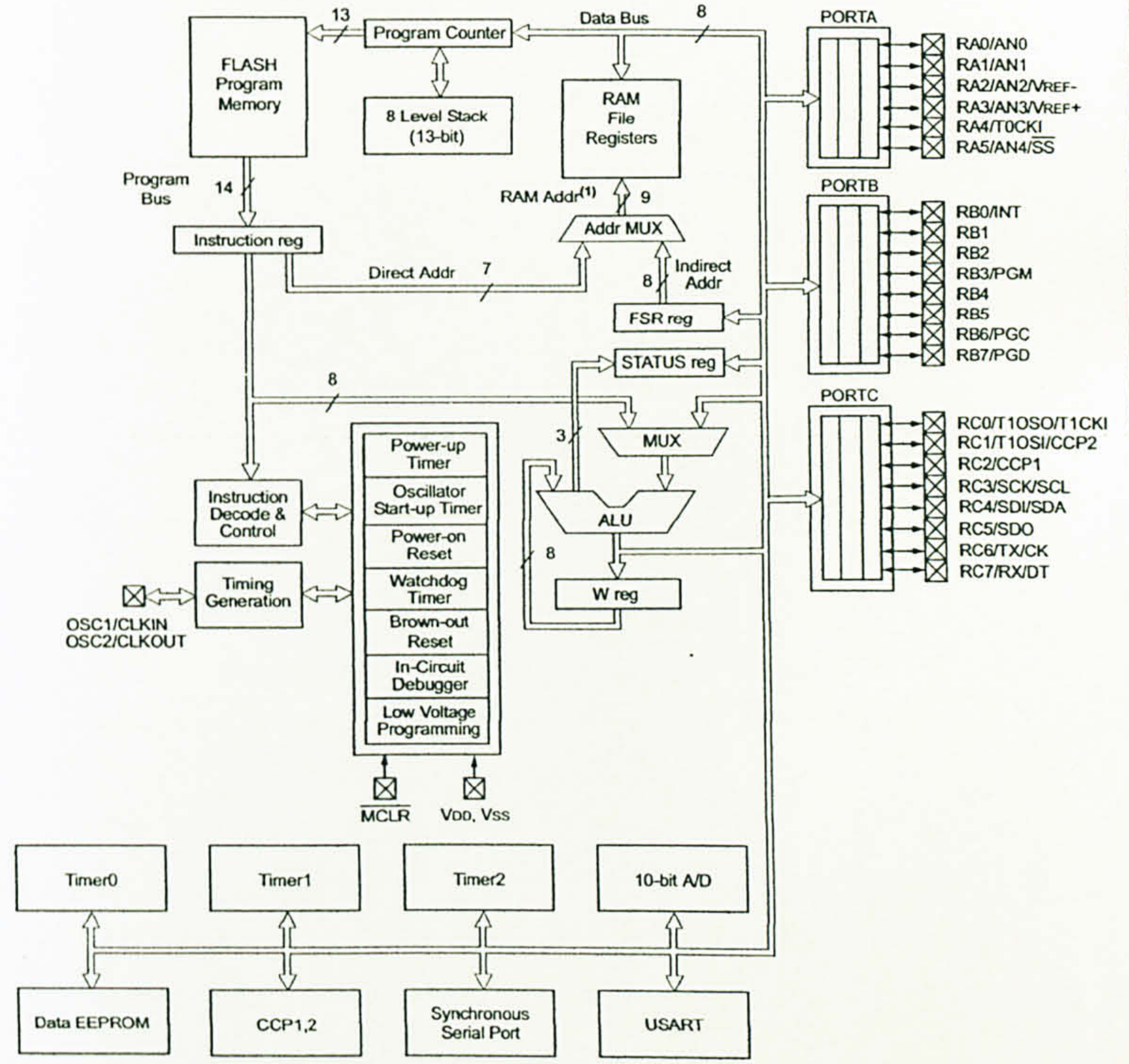
This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

**FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM**

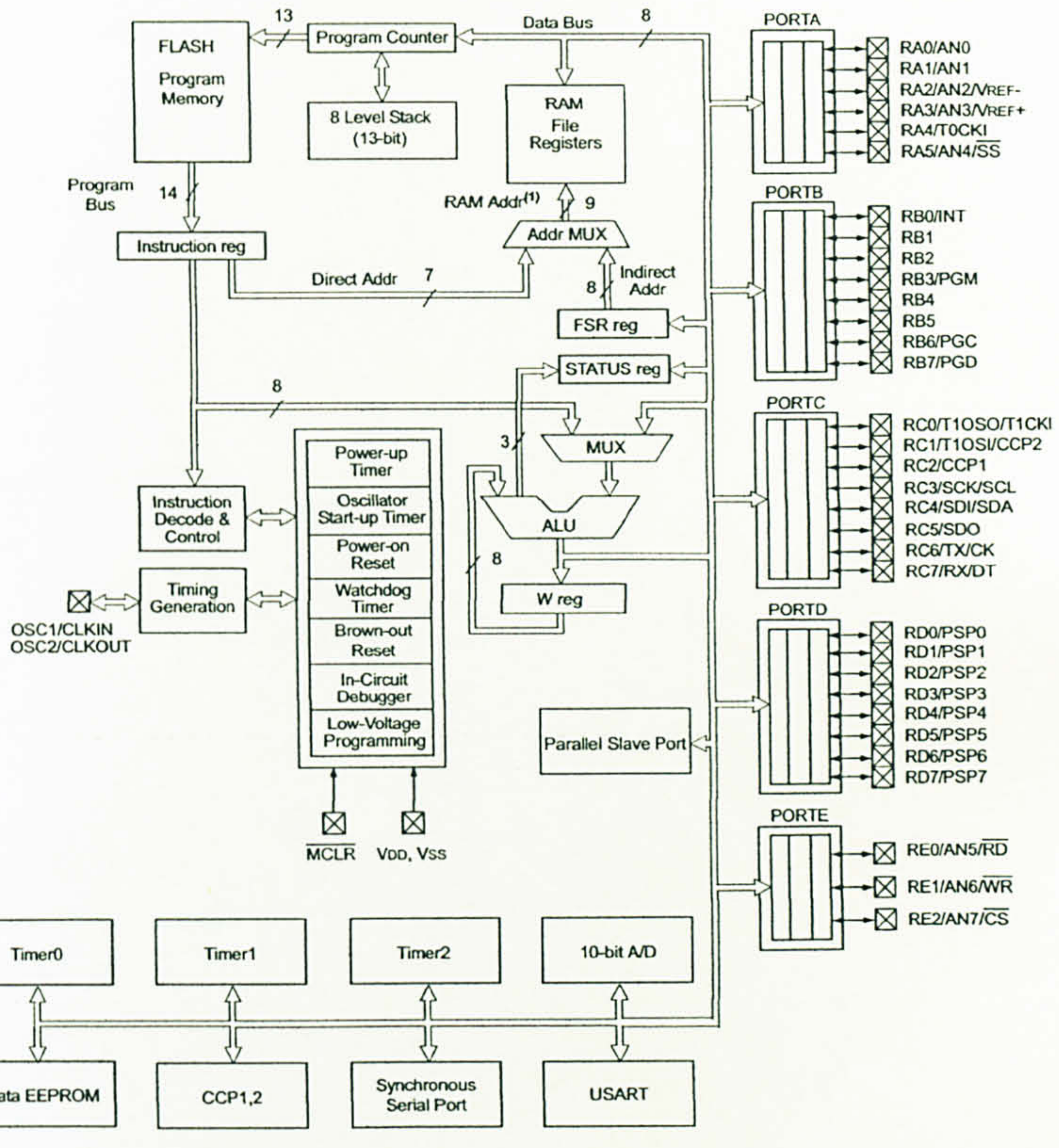
| Device | Program FLASH | Data Memory | Data EEPROM |
|---|---|---|---|
| PIC16F873 | 4K | 192 Bytes | 128 Bytes |
| PIC16F876 | 8K | 368 Bytes | 256 Bytes |




**Note 1:** Higher order bits are from the STATUS register.

**FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM**

| Device | Program FLASH | Data Memory | Data EEPROM |
|---|---|---|---|
| PIC16F874 | 4K | 192 Bytes | 128 Bytes |
| PIC16F877 | 8K | 368 Bytes | 256 Bytes |




**Note 1:** Higher order bits are from the STATUS register.

**TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION**

| Pin Name | DIP Pin# | SOIC Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|
| OSC1/CLKIN | 9 | 9 | I | ST/CMOS(3) | Oscillator crystal input/external clock source input. |
| OSC2/CLKOUT | 10 | 10 | O | — | Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. |
| $\overline{\text{MCLR}}$/VPP | 1 | 1 | I/P | ST | Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device. |
| | | | | | PORTA is a bi-directional I/O port. |
| RA0/AN0 | 2 | 2 | I/O | TTL | RA0 can also be analog input0. |
| RA1/AN1 | 3 | 3 | I/O | TTL | RA1 can also be analog input1. |
| RA2/AN2/VREF- | 4 | 4 | I/O | TTL | RA2 can also be analog input2 or negative analog reference voltage. |
| RA3/AN3/VREF+ | 5 | 5 | I/O | TTL | RA3 can also be analog input3 or positive analog reference voltage. |
| RA4/T0CKI | 6 | 6 | I/O | ST | RA4 can also be the clock input to the Timer0 module. Output is open drain type. |
| RA5/$\overline{\text{SS}}$/AN4 | 7 | 7 | I/O | TTL | RA5 can also be analog input4 or the slave select for the synchronous serial port. |
| | | | | | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. |
| RB0/INT | 21 | 21 | I/O | TTL/ST(1) | RB0 can also be the external interrupt pin. |
| RB1 | 22 | 22 | I/O | TTL | |
| RB2 | 23 | 23 | I/O | TTL | |
| RB3/PGM | 24 | 24 | I/O | TTL | RB3 can also be the low voltage programming input. |
| RB4 | 25 | 25 | I/O | TTL | Interrupt-on-change pin. |
| RB5 | 26 | 26 | I/O | TTL | Interrupt-on-change pin. |
| RB6/PGC | 27 | 27 | I/O | TTL/ST(2) | Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock. |
| RB7/PGD | 28 | 28 | I/O | TTL/ST(2) | Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data. |
| | | | | | PORTC is a bi-directional I/O port. |
| RC0/T1OSO/T1CKI | 11 | 11 | I/O | ST | RC0 can also be the Timer1 oscillator output or Timer1 clock input. |
| RC1/T1OSI/CCP2 | 12 | 12 | I/O | ST | RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output. |
| RC2/CCP1 | 13 | 13 | I/O | ST | RC2 can also be the Capture1 input/Compare1 output/ PWM1 output. |
| RC3/SCK/SCL | 14 | 14 | I/O | ST | RC3 can also be the synchronous serial clock input/output for both SPI and $I^2C$ modes. |
| RC4/SDI/SDA | 15 | 15 | I/O | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O ($I^2C$ mode). |
| RC5/SDO | 16 | 16 | I/O | ST | RC5 can also be the SPI Data Out (SPI mode). |
| RC6/TX/CK | 17 | 17 | I/O | ST | RC6 can also be the USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | 18 | 18 | I/O | ST | RC7 can also be the USART Asynchronous Receive or Synchronous Data. |
| VSS | 8, 19 | 8, 19 | P | — | Ground reference for logic and I/O pins. |
| VDD | 20 | 20 | P | — | Positive supply for logic and I/O pins. |

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

**Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.
**2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.
**3:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

**TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION**

| Pin Name | DIP Pin# | PLCC Pin# | QFP Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| OSC1/CLKIN | 13 | 14 | 30 | I | ST/CMOS(4) | Oscillator crystal input/external clock source input. |
| OSC2/CLKOUT | 14 | 15 | 31 | O | — | Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. |
| $\overline{\text{MCLR}}$/VPP | 1 | 2 | 18 | I/P | ST | Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device. |
| | | | | | | PORTA is a bi-directional I/O port. |
| RA0/AN0 | 2 | 3 | 19 | I/O | TTL | RA0 can also be analog input0. |
| RA1/AN1 | 3 | 4 | 20 | I/O | TTL | RA1 can also be analog input1. |
| RA2/AN2/VREF- | 4 | 5 | 21 | I/O | TTL | RA2 can also be analog input2 or negative analog reference voltage. |
| RA3/AN3/VREF+ | 5 | 6 | 22 | I/O | TTL | RA3 can also be analog input3 or positive analog reference voltage. |
| RA4/T0CKI | 6 | 7 | 23 | I/O | ST | RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type. |
| RA5/$\overline{\text{SS}}$/AN4 | 7 | 8 | 24 | I/O | TTL | RA5 can also be analog input4 or the slave select for the synchronous serial port. |
| | | | | | | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. |
| RB0/INT | 33 | 36 | 8 | I/O | TTL/ST(1) | RB0 can also be the external interrupt pin. |
| RB1 | 34 | 37 | 9 | I/O | TTL | |
| RB2 | 35 | 38 | 10 | I/O | TTL | |
| RB3/PGM | 36 | 39 | 11 | I/O | TTL | RB3 can also be the low voltage programming input. |
| RB4 | 37 | 41 | 14 | I/O | TTL | Interrupt-on-change pin. |
| RB5 | 38 | 42 | 15 | I/O | TTL | Interrupt-on-change pin. |
| RB6/PGC | 39 | 43 | 16 | I/O | TTL/ST(2) | Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock. |
| RB7/PGD | 40 | 44 | 17 | I/O | TTL/ST(2) | Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data. |

*Legend:* *I = input* *O = output* *I/O = input/output* *P = power*
— = Not used TTL = TTL input ST = Schmitt Trigger input

**Note 1:** This buffer is a Schmitt Trigger input when configured as an external interrupt.
**2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.
**3:** This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
**4:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

**TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)**

| Pin Name | DIP Pin# | PLCC Pin# | QFP Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | | | | | | PORTC is a bi-directional I/O port. |
| RC0/T1OSO/T1CKI | 15 | 16 | 32 | I/O | ST | RC0 can also be the Timer1 oscillator output or a Timer1 clock input. |
| RC1/T1OSI/CCP2 | 16 | 18 | 35 | I/O | ST | RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output. |
| RC2/CCP1 | 17 | 19 | 36 | I/O | ST | RC2 can also be the Capture1 input/Compare1 output/PWM1 output. |
| RC3/SCK/SCL | 18 | 20 | 37 | I/O | ST | RC3 can also be the synchronous serial clock input/output for both SPI and $I^2C$ modes. |
| RC4/SDI/SDA | 23 | 25 | 42 | I/O | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O ($I^2C$ mode). |
| RC5/SDO | 24 | 26 | 43 | I/O | ST | RC5 can also be the SPI Data Out (SPI mode). |
| RC6/TX/CK | 25 | 27 | 44 | I/O | ST | RC6 can also be the USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | 26 | 29 | 1 | I/O | ST | RC7 can also be the USART Asynchronous Receive or Synchronous Data. |
| | | | | | | PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus. |
| RD0/PSP0 | 19 | 21 | 38 | I/O | ST/TTL(3) | |
| RD1/PSP1 | 20 | 22 | 39 | I/O | ST/TTL(3) | |
| RD2/PSP2 | 21 | 23 | 40 | I/O | ST/TTL(3) | |
| RD3/PSP3 | 22 | 24 | 41 | I/O | ST/TTL(3) | |
| RD4/PSP4 | 27 | 30 | 2 | I/O | ST/TTL(3) | |
| RD5/PSP5 | 28 | 31 | 3 | I/O | ST/TTL(3) | |
| RD6/PSP6 | 29 | 32 | 4 | I/O | ST/TTL(3) | |
| RD7/PSP7 | 30 | 33 | 5 | I/O | ST/TTL(3) | |
| | | | | | | PORTE is a bi-directional I/O port. |
| RE0/$\overline{RD}$/AN5 | 8 | 9 | 25 | I/O | ST/TTL(3) | RE0 can also be read control for the parallel slave port, or analog input5. |
| RE1/$\overline{WR}$/AN6 | 9 | 10 | 26 | I/O | ST/TTL(3) | RE1 can also be write control for the parallel slave port, or analog input6. |
| RE2/$\overline{CS}$/AN7 | 10 | 11 | 27 | I/O | ST/TTL(3) | RE2 can also be select control for the parallel slave port, or analog input7. |
| VSS | 12,31 | 13,34 | 6,29 | P | — | Ground reference for logic and I/O pins. |
| VDD | 11,32 | 12,35 | 7,28 | P | — | Positive supply for logic and I/O pins. |
| NC | — | 1,17,28, 40 | 12,13, 33,34 | | — | These pins are not internally connected. These pins should be left unconnected. |

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

**Note 1:** This buffer is a Schmitt Trigger input when configured as an external interrupt.
**2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.
**3:** This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
**4:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

## 3.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

### 3.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

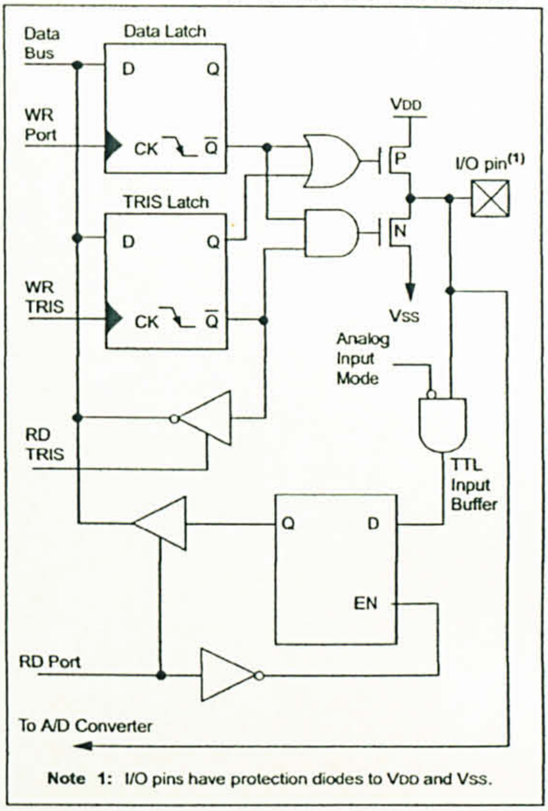
| **Note:** | On a Power-on Reset, these pins are configured as analog inputs and read as '0'. |
|---|---|

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

**EXAMPLE 3-1: INITIALIZING PORTA**

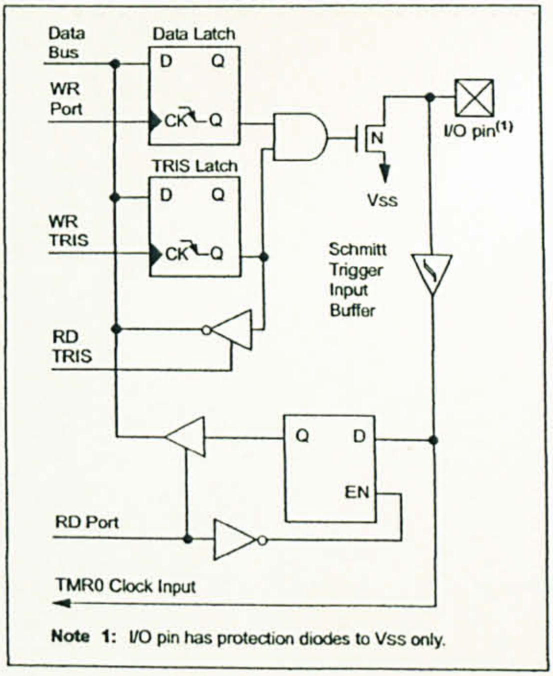
```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

**FIGURE 3-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS**




Note 1: I/O pins have protection diodes to VDD and VSS.

**FIGURE 3-2: BLOCK DIAGRAM OF RA4/T0CKI PIN**




Note 1: I/O pin has protection diodes to VSS only.

**TABLE 3-1: PORTA FUNCTIONS**

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RA0/AN0 | bit0 | TTL | Input/output or analog input. |
| RA1/AN1 | bit1 | TTL | Input/output or analog input. |
| RA2/AN2 | bit2 | TTL | Input/output or analog input. |
| RA3/AN3/VREF | bit3 | TTL | Input/output or analog input or VREF. |
| RA4/T0CKI | bit4 | ST | Input/output or external clock input for Timer0. Output is open drain type. |
| RA5/$\overline{SS}$/AN4 | bit5 | TTL | Input/output or slave select input for synchronous serial port or analog input. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

**TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 05h | PORTA | — | — | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | --0x 0000 | --0u 0000 |
| 85h | TRISA | — | — | PORTA Data Direction Register | | | | | | --11 1111 | --11 1111 |
| 9Fh | ADCON1 | ADFM | — | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | --0- 0000 | --0- 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

**Note:** When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = `0100,0101, 011x, 1101, 1110, 1111`.

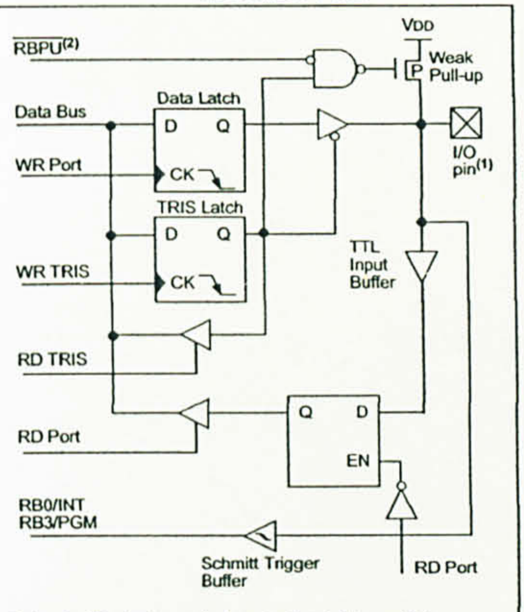
## 3.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the Low Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section.

Each of the PORTB pins has a weak internal pull-up. A *single control bit can turn on all the pull-ups.* This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

**FIGURE 3-3: BLOCK DIAGRAM OF RB3:RB0 PINS**




**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

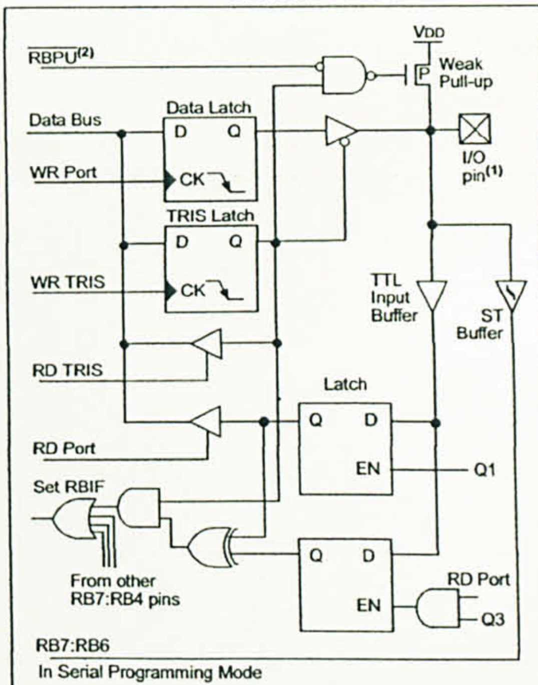
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configureable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the Embedded Control Handbook, *"Implementing Wake-up on Key Strokes"* (AN552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in Section 12.10.1.

**FIGURE 3-4: BLOCK DIAGRAM OF RB7:RB4 PINS**




**Note 1:** I/O pins have diode protection to VDD and VSS.

**2:** To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

### TABLE 3-3: PORTB FUNCTIONS

| Name | Bit# | Buffer | Function |
|---|---|---|---|
| RB0/INT | bit0 | TTL/ST[(1)] | Input/output pin or external interrupt input. Internal software programmable weak pull-up. |
| RB1 | bit1 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB2 | bit2 | TTL | Input/output pin. Internal software programmable weak pull-up. |
| RB3/PGM[(3)] | bit3 | TTL | Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up. |
| RB4 | bit4 | TTL | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. |
| RB5 | bit5 | TTL | Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. |
| RB6/PGC | bit6 | TTL/ST[(2)] | Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock. |
| RB7/PGD | bit7 | TTL/ST[(2)] | Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

**Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.

**2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.

**3:** Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

### TABLE 3-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 06h, 106h | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxx xxxx | uuuu uuuu |
| 86h, 186h | TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| 81h, 181h | OPTION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

## 3.3 PORTC and the TRISC Register

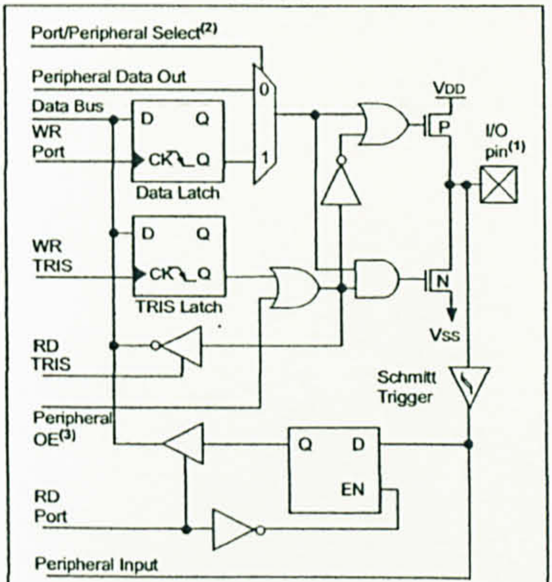
PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 3-5). PORTC pins have Schmitt Trigger input buffers.

When the $I^2C$ module is enabled, the PORTC<4:3> pins can be configured with normal $I^2C$ levels, or with SMBus levels by using the CKE bit (SSPSTAT<6>).

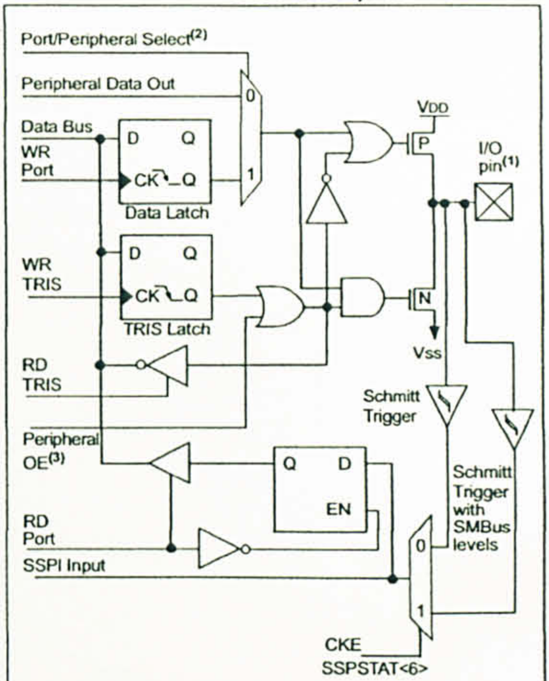
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (`BSF`, `BCF`, `XORWF`) with TRISC as destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

**FIGURE 3-5: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>**




**Note 1:** I/O pins have diode protection to VDD and Vss.
**2:** Port/Peripheral select signal selects between port data and peripheral output.
**3:** Peripheral OE (output enable) is only activated if peripheral select is active.

**FIGURE 3-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>**




**Note 1:** I/O pins have diode protection to VDD and Vss.
**2:** Port/Peripheral select signal selects between port data and peripheral output.
**3:** Peripheral OE (output enable) is only activated if peripheral select is active.

**TABLE 3-5: PORTC FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RC0/T1OSO/T1CKI | bit0 | ST | Input/output port pin or Timer1 oscillator output/Timer1 clock input. |
| RC1/T1OSI/CCP2 | bit1 | ST | Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output. |
| RC2/CCP1 | bit2 | ST | Input/output port pin or Capture1 input/Compare1 output/ PWM1 output. |
| RC3/SCK/SCL | bit3 | ST | RC3 can also be the synchronous serial clock for both SPI and $I^2C$ modes. |
| RC4/SDI/SDA | bit4 | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O ($I^2C$ mode). |
| RC5/SDO | bit5 | ST | Input/output port pin or Synchronous Serial Port data output. |
| RC6/TX/CK | bit6 | ST | Input/output port pin or USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | bit7 | ST | Input/output port pin or USART Asynchronous Receive or Synchronous Data. |

Legend: ST = Schmitt Trigger input

**TABLE 3-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 07h | PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx | uuuu uuuu |
| 87h | TRISC | PORTC Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged

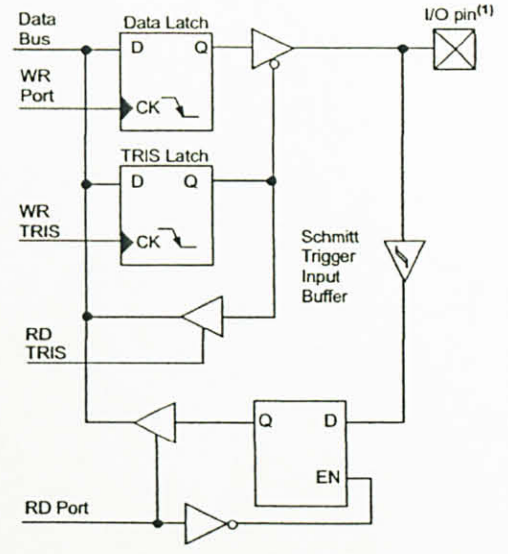
### 3.4 PORTD and TRISD Registers

PORTD and TRISD are not implemented on the PIC16F873 or PIC16F876.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configureable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

**FIGURE 3-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)**




Note 1: I/O pins have protection diodes to VDD and VSS.

**TABLE 3-7: PORTD FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RD0/PSP0 | bit0 | ST/TTL(1) | Input/output port pin or parallel slave port bit0. |
| RD1/PSP1 | bit1 | ST/TTL(1) | Input/output port pin or parallel slave port bit1. |
| RD2/PSP2 | bit2 | ST/TTL(1) | Input/output port pin or parallel slave port bit2. |
| RD3/PSP3 | bit3 | ST/TTL(1) | Input/output port pin or parallel slave port bit3. |
| RD4/PSP4 | bit4 | ST/TTL(1) | Input/output port pin or parallel slave port bit4. |
| RD5/PSP5 | bit5 | ST/TTL(1) | Input/output port pin or parallel slave port bit5. |
| RD6/PSP6 | bit6 | ST/TTL(1) | Input/output port pin or parallel slave port bit6. |
| RD7/PSP7 | bit7 | ST/TTL(1) | Input/output port pin or parallel slave port bit7. |

Legend: ST = Schmitt Trigger input, TTL = TTL input

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

**TABLE 3-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 08h | PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | uuuu uuuu |
| 88h | TRISD | PORTD Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| 89h | TRISE | IBF | OBF | IBOV | PSPMODE | — | PORTE Data Direction Bits | | | 0000 -111 | 0000 -111 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

### 3.5 PORTE and TRISE Register

PORTE and TRISE are not implemented on the PIC16F873 or PIC16F876.

PORTE has three pins (RE0/$\overline{\text{RD}}$/AN5, RE1/$\overline{\text{WR}}$/AN6, and RE2/$\overline{\text{CS}}$/AN7) which are individually configureable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set, and that the pins are configured as digital inputs. Also ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

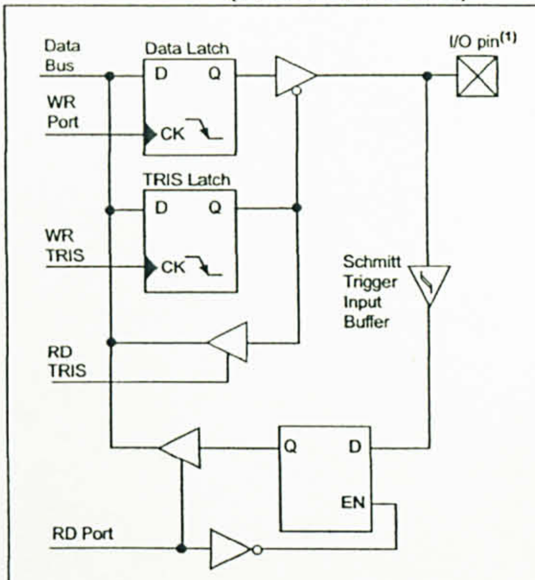
Register 3-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

| Note: | On a Power-on Reset, these pins are configured as analog inputs, and read as '0'. |
|---|---|

**FIGURE 3-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)**




**Note 1:** I/O pins have protection diodes to VDD and VSS.

**TABLE 3-9: PORTE FUNCTIONS**

| Name | Bit# | Buffer Type | Function |
|---|---|---|---|
| RE0/$\overline{\text{RD}}$/AN5 | bit0 | ST/TTL(1) | I/O port pin or read control input in Parallel Slave Port mode or analog input:<br>$\overline{\text{RD}}$<br>1 = Idle<br>0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected) |
| RE1/$\overline{\text{WR}}$/AN6 | bit1 | ST/TTL(1) | I/O port pin or write control input in Parallel Slave Port mode or analog input:<br>$\overline{\text{WR}}$<br>1 = Idle<br>0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected) |
| RE2/$\overline{\text{CS}}$/AN7 | bit2 | ST/TTL(1) | I/O port pin or chip select control input in Parallel Slave Port mode or analog input:<br>$\overline{\text{CS}}$<br>1 = Device is not selected<br>0 = Device is selected |

Legend: ST = Schmitt Trigger input, TTL = TTL input

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

**TABLE 3-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 09h | PORTE | — | — | — | — | — | RE2 | RE1 | RE0 | ---- -xxx | ---- -uuu |
| 89h | TRISE | IBF | OBF | IBOV | PSPMODE | — | PORTE Data Direction Bits | | | 0000 -111 | 0000 -111 |
| 9Fh | ADCON1 | ADFM | — | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | --0- 0000 | --0- 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

**REGISTER 3-1: TRISE REGISTER (ADDRESS 89h)**

| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| IBF | OBF | IBOV | PSPMODE | — | Bit2 | Bit1 | Bit0 |
| bit 7 | | | | | | | bit 0 |

<u>**Parallel Slave Port Status/Control Bits:**</u>

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit
1 = PORTD functions in Parallel Slave Port mode
0 = PORTD functions in general purpose I/O mode

bit 3 **Unimplemented**: Read as '0'

<u>**PORTE Data Direction Bits:**</u>

bit 2 **Bit2**: Direction Control bit for pin RE2/$\overline{CS}$/AN7
1 = Input
0 = Output

bit 1 **Bit1**: Direction Control bit for pin RE1/$\overline{WR}$/AN6
1 = Input
0 = Output

bit 0 **Bit0**: Direction Control bit for pin RE0/$\overline{RD}$/AN5
1 = Input
0 = Output

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

## 5.2 Using Timer0 with an External Clock

When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks. Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to the electrical specification of the desired device.

## 5.3 Prescaler

There is only one prescaler available, which is mutually exclusively shared between the Timer0 module and the Watchdog Timer. A prescaler assignment for the Timer0 module means that there is no prescaler for the Watchdog Timer, and vice-versa. This prescaler is not readable or writable (see Figure 5-1).

The PSA and PS2:PS0 bits (OPTION_REG<3:0>) determine the prescaler assignment and prescale ratio.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g. `CLRF 1, MOVWF 1, BSF 1,x`....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

| **Note:** | Writing to TMR0, when the prescaler is assigned to Timer0, will clear the prescaler count, but will not change the prescaler assignment. |
|---|---|

**REGISTER 5-1: OPTION_REG REGISTER**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RBPU}}$ | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| bit 7 | | | | | | | bit 0 |

bit 7 **$\overline{\text{RBPU}}$**

bit 6 **INTEDG**

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

| Bit Value | TMR0 Rate | WDT Rate |
|---|---|---|
| 000 | 1 : 2 | 1 : 1 |
| 001 | 1 : 4 | 1 : 2 |
| 010 | 1 : 8 | 1 : 4 |
| 011 | 1 : 16 | 1 : 8 |
| 100 | 1 : 32 | 1 : 16 |
| 101 | 1 : 64 | 1 : 32 |
| 110 | 1 : 128 | 1 : 64 |
| 111 | 1 : 256 | 1 : 128 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

| **Note:** | To avoid an unintended device RESET, the instruction sequence shown in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled. |
|---|---|

**TABLE 5-1: REGISTERS ASSOCIATED WITH TIMER0**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other RESETS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 01h,101h | TMR0 | Timer0 Module's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Bh,8Bh, 10Bh,18Bh | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 0000 000x | 0000 000u |
| 81h,181h | OPTION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by Timer0.

## APPENDIX D

## L78XX VOLTAGE REGULATOR DATASHEET

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

# KA78XX/KA78XXA

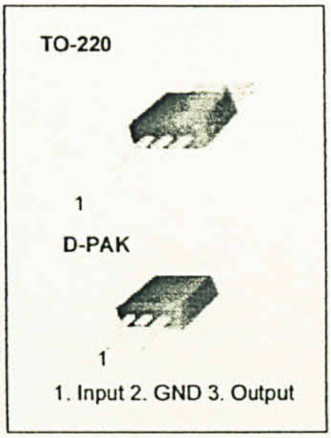
## 3-Terminal 1A Positive Voltage Regulator

### Features

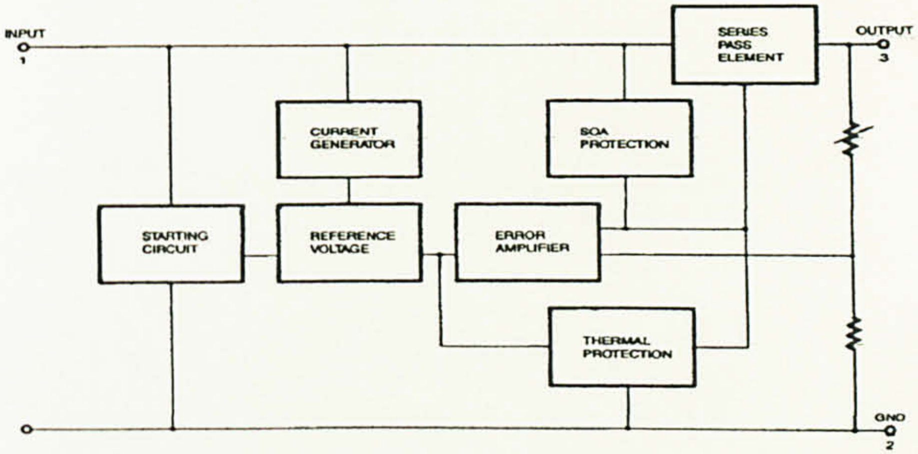
- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

### Description

The KA78XX/KA78XXA series of three-terminal positive regulator are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.




### Internal Block Digram




Rev. 1.0.0

## Absolute Maximum Ratings

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Input Voltage (for $V_O$ = 5V to 18V)<br>(for $V_O$ = 24V) | $V_I$<br>$V_I$ | 35<br>40 | V<br>V |
| Thermal Resistance Junction-Cases (TO-220) | $R_{\theta JC}$ | 5 | °C/W |
| Thermal Resistance Junction-Air (TO-220) | $R_{\theta JA}$ | 65 | °C/W |
| Operating Temperature Range (KA78XX/A/R) | $T_{OPR}$ | 0 ~ +125 | °C |
| Storage Temperature Range | $T_{STG}$ | -65 ~ +150 | °C |

## Electrical Characteristics (KA7805/KA7805R)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =10V, $C_I$= 0.33μF, $C_O$=0.1μF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7805 Min. | KA7805 Typ. | KA7805 Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 4.8 | 5.0 | 5.2 | V |
| | | 5.0mA ≤ Io ≤ 1.0A, $P_O$ ≤ 15W<br>$V_I$ = 7V to 20V | | 4.75 | 5.0 | 5.25 | |
| Line Regulation (Note1) | Regline | $T_J$=+25 °C | $V_O$ = 7V to 25V | - | 4.0 | 100 | mV |
| | | | $V_I$ = 8V to 12V | - | 1.6 | 50 | |
| Load Regulation (Note1) | Regload | $T_J$=+25 °C | $I_O$ = 5.0mA to1.5A | - | 9 | 100 | mV |
| | | | $I_O$ =250mA to 750mA | - | 4 | 50 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.0 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | 0.03 | 0.5 | mA |
| | | $V_I$= 7V to 25V | | - | 0.3 | 1.3 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$= 5mA | | - | -0.8 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$=+25 °C | | - | 42 | - | μV/$V_O$ |
| Ripple Rejection | RR | f = 120Hz<br>$V_O$ = 8V to 18V | | 62 | 73 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$ =+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 15 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$ =+25 °C | | - | 230 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_o$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7806/KA7806R)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =11V, $C_I$= 0.33μF, $C_O$=0.1μF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7806 Min. | KA7806 Typ. | KA7806 Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 5.75 | 6.0 | 6.25 | V |
| | | 5.0mA ≤ $I_O$ ≤ 1.0A, $P_O$ ≤ 15W<br>$V_I$ = 8.0V to 21V | | 5.7 | 6.0 | 6.3 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 8V to 25V | - | 5 | 120 | mV |
| | | | $V_I$ = 9V to 13V | - | 1.5 | 60 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ =5mA to 1.5A | - | 9 | 120 | mV |
| | | | $I_O$ =250mA to750mA | - | 3 | 60 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.0 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1A | | - | - | 0.5 | mA |
| | | $V_I$ = 8V to 25V | | - | - | 1.3 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -0.8 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 45 | - | μV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 9V to 19V | | 59 | 75 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$ =+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 19 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$= 35V, $T_A$=+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7808/KA7808R)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =14V, $C_I$= 0.33μF, $C_O$=0.1μF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7808 | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | | Min. | Typ. | Max. | |
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 7.7 | 8.0 | 8.3 | V |
| | | 5.0mA ≤ $I_O$ ≤ 1.0A, $P_O$ ≤ 15W<br>$V_I$ = 10.5V to 23V | | 7.6 | 8.0 | 8.4 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 10.5V to 25V | - | 5.0 | 160 | mV |
| | | | $V_I$ = 11.5V to 17V | - | 2.0 | 80 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5.0mA to 1.5A | - | 10 | 160 | mV |
| | | | $I_O$= 250mA to 750mA | - | 5.0 | 80 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.0 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | 0.05 | 0.5 | mA |
| | | $V_I$ = 10.5A to 25V | | - | 0.5 | 1.0 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -0.8 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 52 | - | μV/Vo |
| Ripple Rejection | RR | f = 120Hz, $V_I$= 11.5V to 21.5V | | 56 | 73 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 17 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$= 35V, $T_A$ =+25 °C | | - | 230 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7809/KA7809R)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =15V, $C_I$= 0.33µF, $C_O$=0.1µF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7809 Min. | KA7809 Typ. | KA7809 Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 8.65 | 9 | 9.35 | V |
| | | 5.0mA≤ $I_O$ ≤1.0A, $P_O$ ≤15W<br>$V_I$= 11.5V to 24V | | 8.6 | 9 | 9.4 | |
| Line Regulation (Note1) | Regline | $T_J$=+25 °C | $V_I$ = 11.5V to 25V | - | 6 | 180 | mV |
| | | | $V_I$ = 12V to 17V | - | 2 | 90 | |
| Load Regulation (Note1) | Regload | $T_J$=+25 °C | $I_O$ = 5mA to 1.5A | - | 12 | 180 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 4 | 90 | |
| Quiescent Current | $I_Q$ | $T_J$=+25 °C | | - | 5.0 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | - | 0.5 | mA |
| | | $V_I$ = 11.5V to 26V | | - | - | 1.3 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 58 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 13V to 23V | | 56 | 71 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 17 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$= 35V, $T_A$ =+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$= +25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7810)

(Refer to test circuit ,$0^{\circ}C < T_J < 125^{\circ}C$, $I_O$ = 500mA, $V_I$ =16V, $C_I$= 0.33µF, $C_O$=0.1µF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7810 Min. | KA7810 Typ. | KA7810 Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 9.6 | 10 | 10.4 | V |
| | | 5.0mA ≤ $I_O$ ≤ 1.0A, $P_O$ ≤ 15W<br>$V_I$ = 12.5V to 25V | | 9.5 | 10 | 10.5 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 12.5V to 25V | - | 10 | 200 | mV |
| | | | $V_I$ = 13V to 25V | - | 3 | 100 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5mA to 1.5A | - | 12 | 200 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 4 | 400 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.1 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | - | 0.5 | mA |
| | | $V_I$ = 12.5V to 29V | | - | - | 1.0 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 58 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 13V to 23V | | 56 | 71 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 17 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$=+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7812/KA7812R)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =19V, $C_I$= 0.33µF, $C_O$=0.1µF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7812/KA7812R Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 11.5 | 12 | 12.5 | V |
| | | 5.0mA ≤ $I_O$≤1.0A, $P_O$≤15W<br>$V_I$ = 14.5V to 27V | | 11.4 | 12 | 12.6 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 14.5V to 30V | - | 10 | 240 | mV |
| | | | $V_I$ = 16V to 22V | - | 3.0 | 120 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5mA to 1.5A | - | 11 | 240 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 5.0 | 120 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.1 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | 0.1 | 0.5 | mA |
| | | $V_I$ = 14.5V to 30V | | - | 0.5 | 1.0 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 76 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 15V to 25V | | 55 | 71 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 18 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$=+25 °C | | - | 230 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ = +25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7815)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =23V, $C_I$= 0.33µF, $C_O$=0.1µF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7815 Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 14.4 | 15 | 15.6 | V |
| | | 5.0mA ≤ $I_O$≤1.0A, $P_O$≤15W<br>$V_I$ = 17.5V to 30V | | 14.25 | 15 | 15.75 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 17.5V to 30V | - | 11 | 300 | mV |
| | | | $V_I$ = 20V to 26V | - | 3 | 150 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5mA to 1.5A | - | 12 | 300 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 4 | 150 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.2 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | - | 0.5 | mA |
| | | $V_I$ = 17.5V to 30V | | - | - | 1.0 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 90 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 18.5V to 28.5V | | 54 | 70 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 19 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$=+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7818)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =27V, $C_I$= 0.33μF, $C_O$=0.1μF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7818 Min. | KA7818 Typ. | KA7818 Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 17.3 | 18 | 18.7 | V |
| | | 5.0mA ≤ $I_O$ ≤1.0A, $P_O$ ≤15W<br>$V_I$ = 21V to 33V | | 17.1 | 18 | 18.9 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 21V to 33V | - | 15 | 360 | mV |
| | | | $V_I$ = 24V to 30V | - | 5 | 180 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5mA to 1.5A | - | 15 | 360 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 5.0 | 180 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.2 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | - | 0.5 | mA |
| | | $V_I$ = 21V to 33V | | - | - | 1 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 110 | - | μV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 22V to 32V | | 53 | 69 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 22 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$=+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7824)

(Refer to test circuit ,0°C < $T_J$ < 125°C, $I_O$ = 500mA, $V_I$ =33V, $C_I$= 0.33μF, $C_O$=0.1μF, unless otherwise specified)

| Parameter | Symbol | Conditions | | KA7824 Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 23 | 24 | 25 | V |
| | | 5.0mA ≤ $I_O$ ≤ 1.0A, $P_O$ ≤ 15W<br>$V_I$ = 27V to 38V | | 22.8 | 24 | 25.25 | |
| Line Regulation (Note1) | Regline | $T_J$ =+25 °C | $V_I$ = 27V to 38V | - | 17 | 480 | mV |
| | | | $V_I$ = 30V to 36V | - | 6 | 240 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C | $I_O$ = 5mA to 1.5A | - | 15 | 480 | mV |
| | | | $I_O$ = 250mA to 750mA | - | 5.0 | 240 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 5.2 | 8.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1.0A | | - | 0.1 | 0.5 | mA |
| | | $V_I$ = 27V to 38V | | - | 0.5 | 1 | |
| Output Voltage Drift | $\Delta V_O/\Delta T$ | $I_O$ = 5mA | | - | -1.5 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz, $T_A$ =+25 °C | | - | 60 | - | μV/Vo |
| Ripple Rejection | RR | f = 120Hz<br>$V_I$ = 28V to 38V | | 50 | 67 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$=+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 28 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$ = 35V, $T_A$=+25 °C | | - | 230 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$ =+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7805A)

(Refer to the test circuits. $0^{o}C < T_J < +125^{o}C$, $I_o = 1A$, $V_I = 10V$, $C_I = 0.33\mu F$, $C_O = 0.1\mu F$, unless otherwise specified)

| Parameter | Symbol | Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J = +25^{o}C$ | | 4.9 | 5 | 5.1 | V |
| | | $I_O$ = 5mA to 1A, $P_O \leq 15W$ $V_I$ = 7.5V to 20V | | 4.8 | 5 | 5.2 | |
| Line Regulation (Note1) | Regline | $V_I$ = 7.5V to 25V $I_O$ = 500mA | | - | 5 | 50 | mV |
| | | $V_I$ = 8V to 12V | | - | 3 | 50 | |
| | | $T_J = +25^{o}C$ | $V_I$= 7.3V to 20V | - | 5 | 50 | |
| | | | $V_I$= 8V to 12V | - | 1.5 | 25 | |
| Load Regulation (Note1) | Regload | $T_J = +25^{o}C$ $I_O$ = 5mA to 1.5A | | - | 9 | 100 | mV |
| | | $I_O$ = 5mA to 1A | | - | 9 | 100 | |
| | | $I_O$ = 250mA to 750mA | | - | 4 | 50 | |
| Quiescent Current | $I_Q$ | $T_J = +25^{o}C$ | | - | 5.0 | 6.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1A | | - | - | 0.5 | mA |
| | | $V_I$ = 8 V to 25V, $I_O$ = 500mA | | - | - | 0.8 | |
| | | $V_I$ = 7.5V to 20V, $T_J = +25^{o}C$ | | - | - | 0.8 | |
| Output Voltage Drift | $\Delta V/\Delta T$ | Io = 5mA | | - | -0.8 | - | mV/ $^{o}C$ |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz $T_A = +25^{o}C$ | | - | 10 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz, $I_O$ = 500mA $V_I$ = 8V to 18V | | - | 68 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J = +25^{o}C$ | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 17 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$= 35V, $T_A = +25^{o}C$ | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J = +25^{o}C$ | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Change in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

## Electrical Characteristics (KA7806A)

(Refer to the test circuits. $0^{o}C < T_J < +125^{o}C$, $I_o$ =1A, $V_I$ = 11V, $C_I$=0.33µF, $C_O$=0.1µF, unless otherwise specified)

| Parameter | Symbol | Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| Output Voltage | $V_O$ | $T_J$ =+25 °C | | 5.58 | 6 | 6.12 | V |
| | | $I_O$ = 5mA to 1A, $P_O \leq$ 15W<br>$V_I$ = 8.6V to 21V | | 5.76 | 6 | 6.24 | |
| Line Regulation (Note1) | Regline | $V_I$= 8.6V to 25V<br>$I_O$ = 500mA | | - | 5 | 60 | mV |
| | | $V_I$= 9V to 13V | | - | 3 | 60 | |
| | | $T_J$ =+25 °C | $V_I$= 8.3V to 21V | - | 5 | 60 | |
| | | | $V_I$= 9V to 13V | - | 1.5 | 30 | |
| Load Regulation (Note1) | Regload | $T_J$ =+25 °C<br>$I_O$ = 5mA to 1.5A | | - | 9 | 100 | mV |
| | | $I_O$ = 5mA to 1A | | - | 4 | 100 | |
| | | $I_O$ = 250mA to 750mA | | - | 5.0 | 50 | |
| Quiescent Current | $I_Q$ | $T_J$ =+25 °C | | - | 4.3 | 6.0 | mA |
| Quiescent Current Change | $\Delta I_Q$ | $I_O$ = 5mA to 1A | | - | - | 0.5 | mA |
| | | $V_I$ = 9V to 25V, $I_O$ = 500mA | | - | - | 0.8 | |
| | | $V_I$= 8.5V to 21V, $T_J$ =+25 °C | | - | - | 0.8 | |
| Output Voltage Drift | $\Delta V/\Delta T$ | $I_O$ = 5mA | | - | -0.8 | - | mV/ °C |
| Output Noise Voltage | $V_N$ | f = 10Hz to 100KHz<br>$T_A$ =+25 °C | | - | 10 | - | µV/Vo |
| Ripple Rejection | RR | f = 120Hz, $I_O$ = 500mA<br>$V_I$ = 9V to 19V | | - | 65 | - | dB |
| Dropout Voltage | $V_{Drop}$ | $I_O$ = 1A, $T_J$ =+25 °C | | - | 2 | - | V |
| Output Resistance | $r_O$ | f = 1KHz | | - | 17 | - | mΩ |
| Short Circuit Current | $I_{SC}$ | $V_I$= 35V, $T_A$ =+25 °C | | - | 250 | - | mA |
| Peak Current | $I_{PK}$ | $T_J$=+25 °C | | - | 2.2 | - | A |

**Note:**

1. Load and line regulation are specified at constant junction temperature. Change in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.

# Typical Perfomance Characteristics

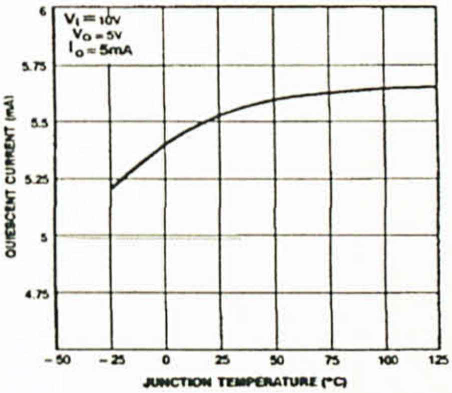



Figure 1. Quiescent Current

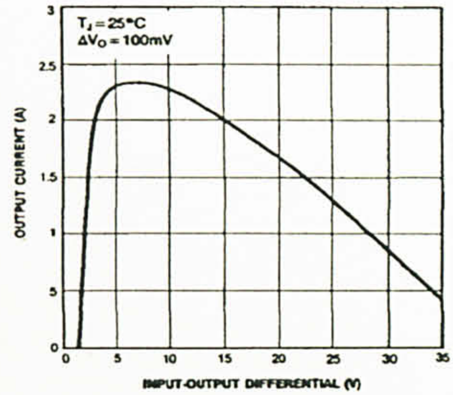



Figure 2. Peak Output Current

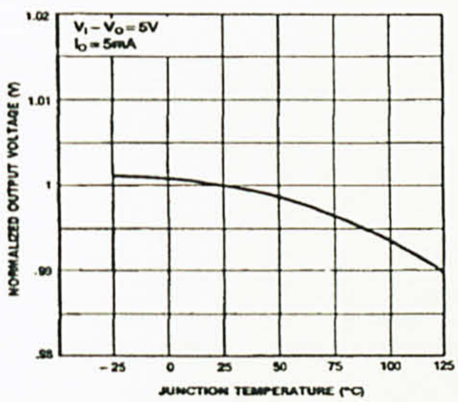



Figure 3. Output Voltage

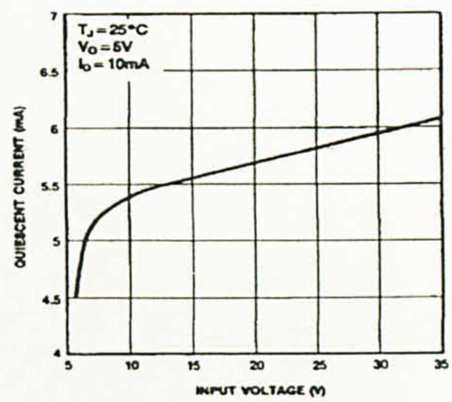



Figure 4. Quiescent Current

# Typical Applications

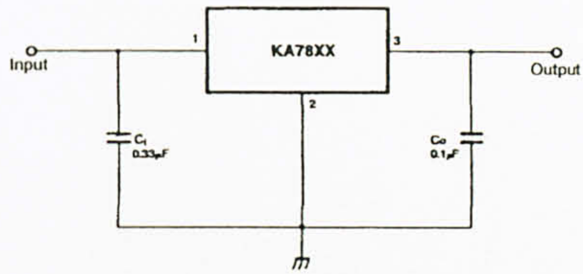



Figure 5. DC Parameters

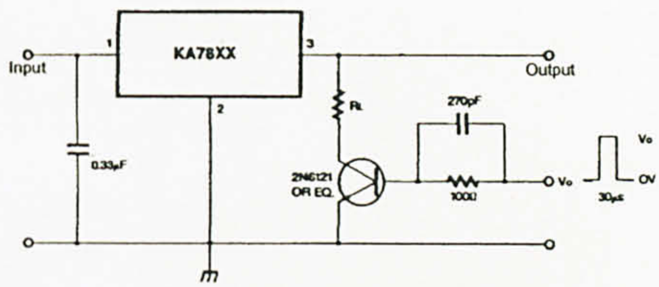



Figure 6. Load Regulation

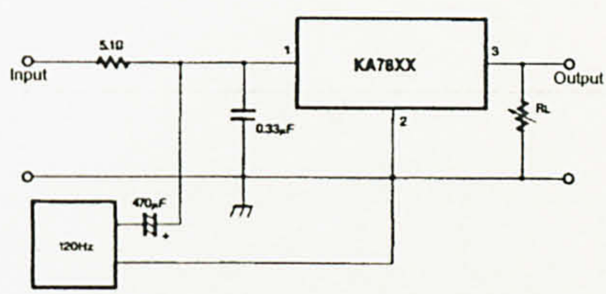



Figure 7. Ripple Rejection

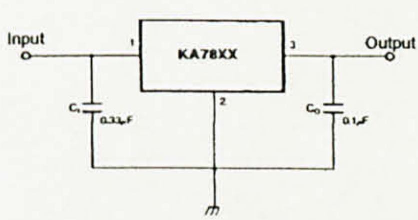
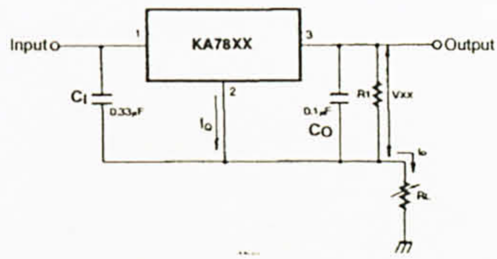



Figure 8. Fixed Output Regulator




$$I_O = \frac{V_{XX}}{R_1} + I_Q$$

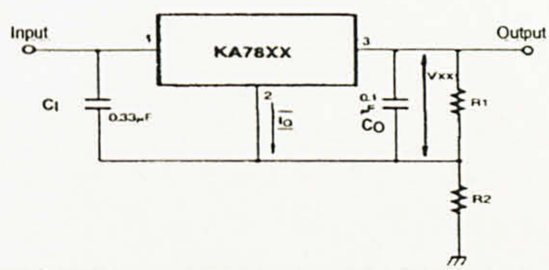
**Figure 9. Constant Current Regulator**

**Notes:**

(1) To specify an output voltage. substitute voltage value for "XX." A common ground is required between the input and the Output voltage. The input voltage must remain typically 2.0V above the output voltage even during the low point on the input ripple voltage.

(2) $C_I$ is required if regulator is located an appreciable distance from power Supply filter.

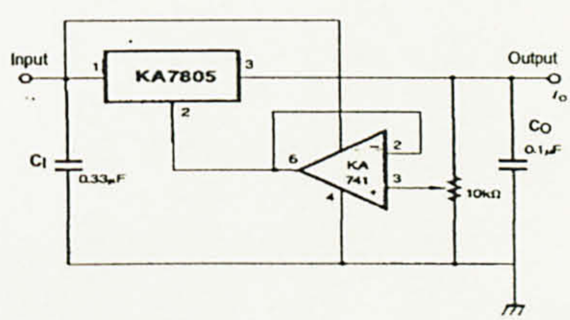
(3) $C_O$ improves stability and transient response.




$I_{RI} \geq 5I_Q$

$V_O = V_{XX}(1+R_2/R_1)+I_QR_2$

**Figure 10. Circuit for Increasing Output Voltage**




$I_{RI} \geq 5\ I_Q$

$V_O = V_{XX}(1+R_2/R_1)+I_QR_2$

**Figure 11. Adjustable Output Regulator (7 to 30V)**

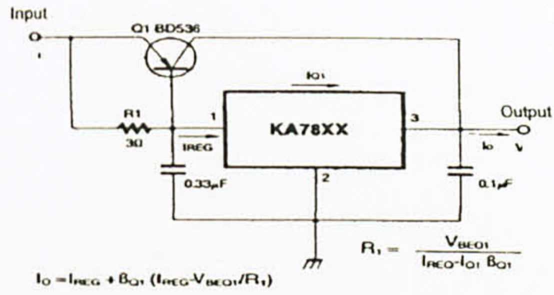



Figure 12. High Current Voltage Regulator

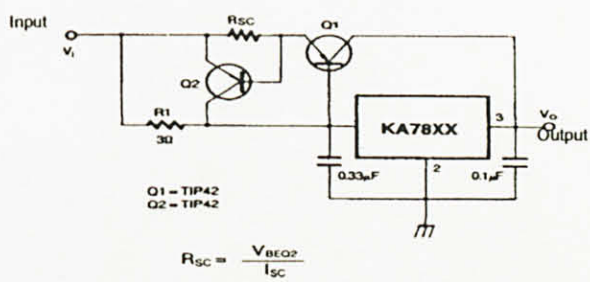



Figure 13. High Output Current with Short Circuit Protection

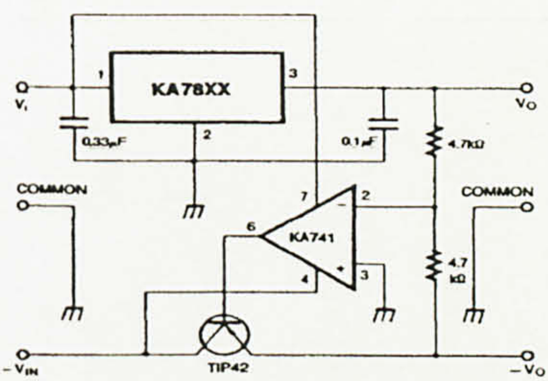
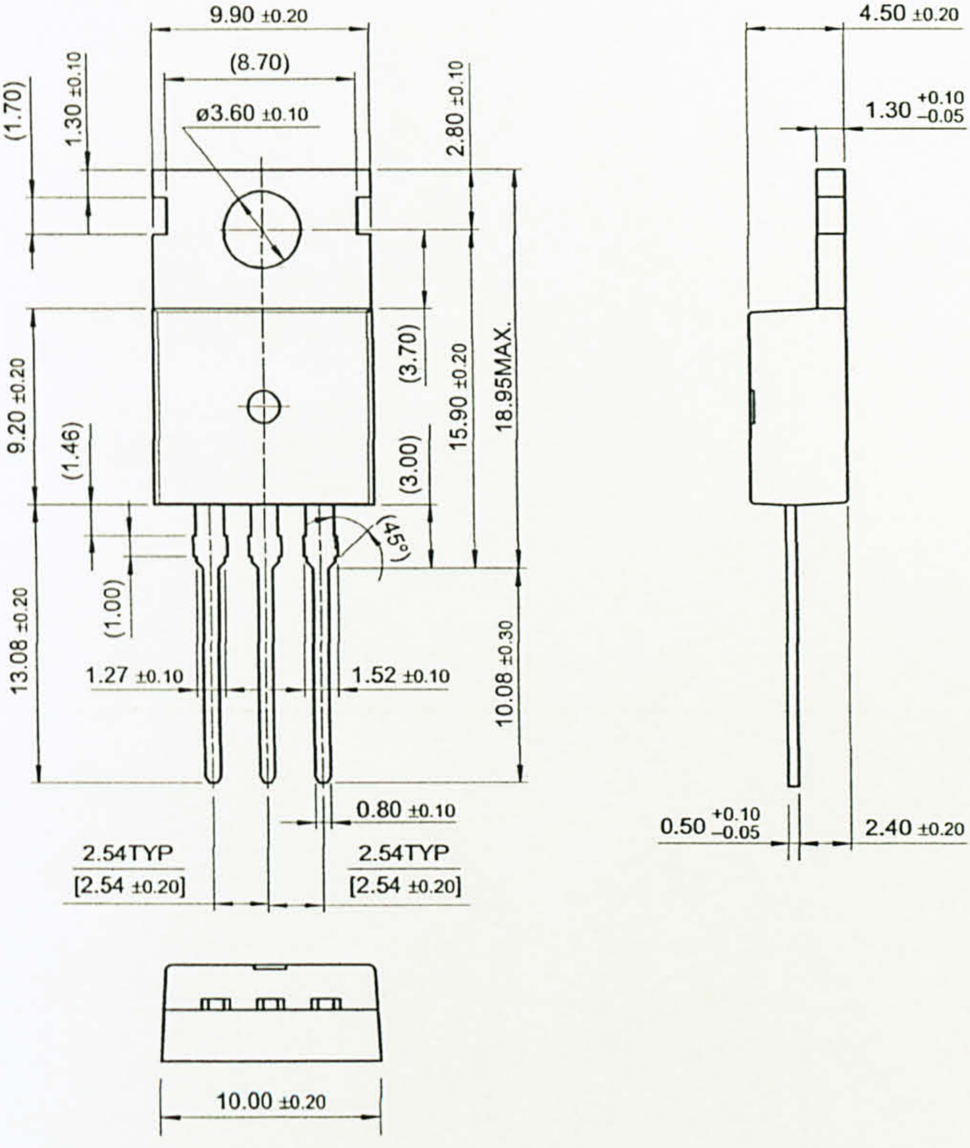



Figure 14. Tracking Voltage Regulator

## Mechanical Dimensions

### Package

# TO-220



## Ordering Information

| Product Number | Output Voltage Tolerance | Package | Operating Temperature |
|---|---|---|---|
| KA7805 / KA7806 | ±4% | TO-220 | 0 ~ + 125°C |
| KA7808 / KA7809 | | | |
| KA7810 | | | |
| KA7812 / KA7815 | | | |
| KA7818 / KA7824 | | | |
| KA7805A / KA7806A | ±2% | | |
| KA7808A / KA7809A | | | |
| KA7810A / KA7812A | | | |
| KA7815A / KA7818A | | | |
| KA7824A | | | |
| KA7805R / KA7806R | ±4% | D-PAK | |
| KA7808R / KA7809R | | | |
| KA7812R | | | |

## APPENDIX E

## L298N H-BRIDGE DRIVER DATASHEET

# DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

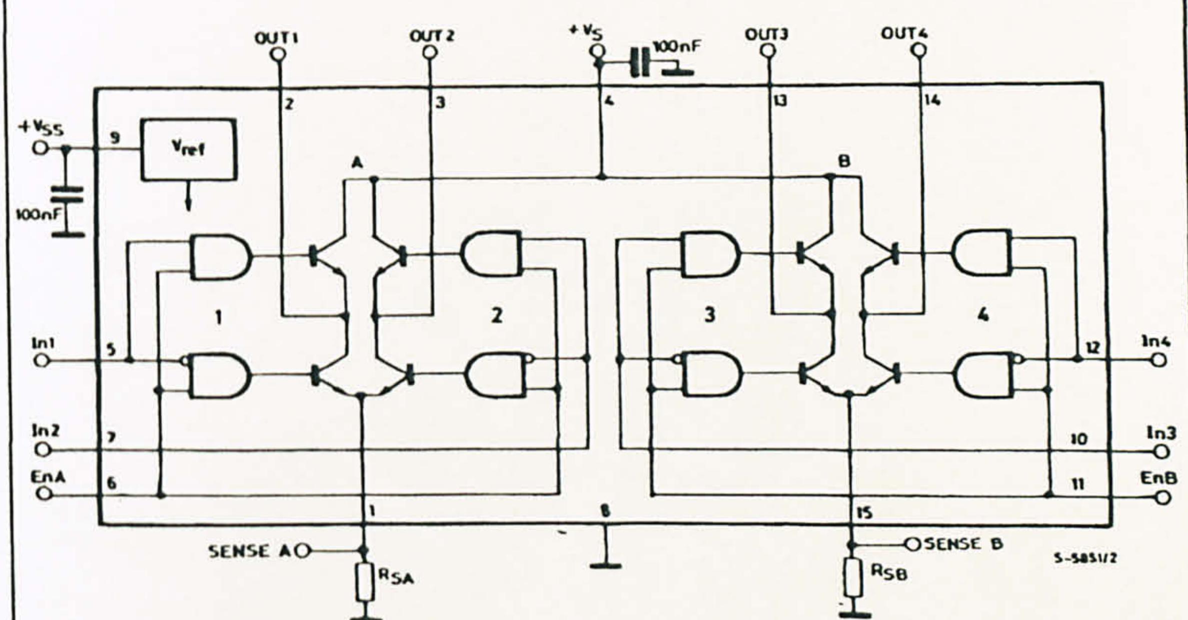
Multiwatt15　　PowerSO20

**ORDERING NUMBERS :** L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

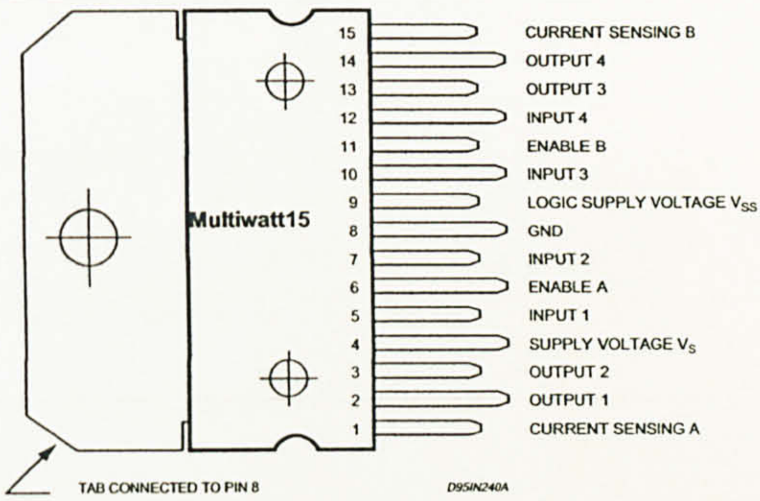
### BLOCK DIAGRAM

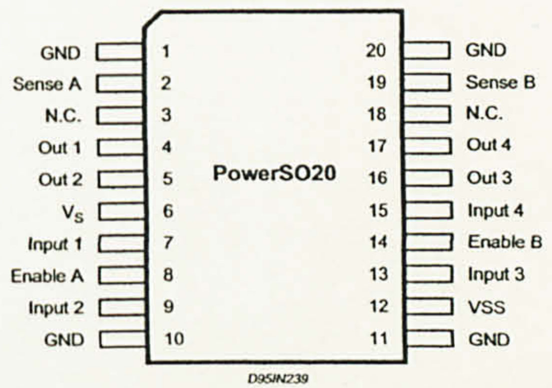


## ABSOLUTE MAXIMUM RATINGS

| Symbol | Parameter | Value | Unit |
|---|---|---|---|
| $V_S$ | Power Supply | 50 | V |
| $V_{SS}$ | Logic Supply Voltage | 7 | V |
| $V_I, V_{en}$ | Input and Enable Voltage | –0.3 to 7 | V |
| $I_O$ | Peak Output Current (each Channel)<br>– Non Repetitive (t = 100μs)<br>–Repetitive (80% on –20% off; $t_{on}$ = 10ms)<br>–DC Operation | <br>3<br>2.5<br>2 | <br>A<br>A<br>A |
| $V_{sens}$ | Sensing Voltage | –1 to 2.3 | V |
| $P_{tot}$ | Total Power Dissipation ($T_{case}$ = 75°C) | 25 | W |
| $T_{op}$ | Junction Operating Temperature | –25 to 130 | °C |
| $T_{stg}, T_j$ | Storage and Junction Temperature | –40 to 150 | °C |

## PIN CONNECTIONS (top view)




## THERMAL DATA

| Symbol | Parameter | | PowerSO20 | Multiwatt15 | Unit |
|---|---|---|---|---|---|
| $R_{th\ j\text{-}case}$ | Thermal Resistance Junction-case | Max. | – | 3 | °C/W |
| $R_{th\ j\text{-}amb}$ | Thermal Resistance Junction-ambient | Max. | 13 (*) | 35 | °C/W |

(*) Mounted on aluminum substrate

**PIN FUNCTIONS** *(refer to the block diagram)*

| MW.15 | PowerSO | Name | Function |
|---|---|---|---|
| 1;15 | 2;19 | Sense A; Sense B | Between this pin and ground is connected the sense resistor to control the current of the load. |
| 2;3 | 4;5 | Out 1; Out 2 | Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1. |
| 4 | 6 | $V_S$ | Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground. |
| 5;7 | 7;9 | Input 1; Input 2 | TTL Compatible Inputs of the Bridge A. |
| 6;11 | 8;14 | Enable A; Enable B | TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B). |
| 8 | 1,10,11,20 | GND | Ground. |
| 9 | 12 | VSS | Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground. |
| 10; 12 | 13;15 | Input 3; Input 4 | TTL Compatible Inputs of the Bridge B. |
| 13; 14 | 16;17 | Out 3; Out 4 | Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15. |
| – | 3;18 | N.C. | Not Connected |

**ELECTRICAL CHARACTERISTICS** ($V_S$ = 42V; $V_{SS}$ = 5V, $T_j$ = 25°C; unless otherwise specified)

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_S$ | Supply Voltage (pin 4) | Operative Condition | $V_{IH}$ +2.5 | | 46 | V |
| $V_{SS}$ | Logic Supply Voltage (pin 9) | | 4.5 | 5 | 7 | V |
| $I_S$ | Quiescent Supply Current (pin 4) | $V_{en}$ = H; $I_L$ = 0 $V_i$ = L<br>$V_i$ = H | | 13<br>50 | 22<br>70 | mA<br>mA |
| | | $V_{en}$ = L $V_i$ = X | | | 4 | mA |
| $I_{SS}$ | Quiescent Current from $V_{SS}$ (pin 9) | $V_{en}$ = H; $I_L$ = 0 $V_i$ = L<br>$V_i$ = H | | 24<br>7 | 36<br>12 | mA<br>mA |
| | | $V_{en}$ = L $V_i$ = X | | | 6 | mA |
| $V_{iL}$ | Input Low Voltage (pins 5, 7, 10, 12) | | –0.3 | | 1.5 | V |
| $V_{iH}$ | Input High Voltage (pins 5, 7, 10, 12) | | 2.3 | | VSS | V |
| $I_{iL}$ | Low Voltage Input Current (pins 5, 7, 10, 12) | $V_i$ = L | | | –10 | µA |
| $I_{iH}$ | High Voltage Input Current (pins 5, 7, 10, 12) | Vi = H ≤ $V_{SS}$ –0.6V | | 30 | 100 | µA |
| $V_{en}$ = L | Enable Low Voltage (pins 6, 11) | | –0.3 | | 1.5 | V |
| $V_{en}$ = H | Enable High Voltage (pins 6, 11) | | 2.3 | | $V_{SS}$ | V |
| $I_{en}$ = L | Low Voltage Enable Current (pins 6, 11) | $V_{en}$ = L | | | –10 | µA |
| $I_{en}$ = H | High Voltage Enable Current (pins 6, 11) | $V_{en}$ = H ≤ $V_{SS}$ –0.6V | | 30 | 100 | µA |
| $V_{CEsat (H)}$ | Source Saturation Voltage | $I_L$ = 1A<br>$I_L$ = 2A | 0.95 | 1.35<br>2 | 1.7<br>2.7 | V<br>V |
| $V_{CEsat (L)}$ | Sink Saturation Voltage | $I_L$ = 1A (5)<br>$I_L$ = 2A (5) | 0.85 | 1.2<br>1.7 | 1.6<br>2.3 | V<br>V |
| $V_{CEsat}$ | Total Drop | $I_L$ = 1A (5)<br>$I_L$ = 2A (5) | 1.80 | | 3.2<br>4.9 | V<br>V |
| $V_{sens}$ | Sensing Voltage (pins 1, 15) | | –1 (1) | | 2 | V |

## ELECTRICAL CHARACTERISTICS (continued)

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $T_1$ ($V_i$) | Source Current Turn-off Delay | 0.5 $V_i$ to 0.9 $I_L$ (2); (4) | | 1.5 | | μs |
| $T_2$ ($V_i$) | Source Current Fall Time | 0.9 $I_L$ to 0.1 $I_L$ (2); (4) | | 0.2 | | μs |
| $T_3$ ($V_i$) | Source Current Turn-on Delay | 0.5 $V_i$ to 0.1 $I_L$ (2); (4) | | 2 | | μs |
| $T_4$ ($V_i$) | Source Current Rise Time | 0.1 $I_L$ to 0.9 $I_L$ (2); (4) | | 0.7 | | μs |
| $T_5$ ($V_i$) | Sink Current Turn-off Delay | 0.5 $V_i$ to 0.9 $I_L$ (3); (4) | | 0.7 | | μs |
| $T_6$ ($V_i$) | Sink Current Fall Time | 0.9 $I_L$ to 0.1 $I_L$ (3); (4) | | 0.25 | | μs |
| $T_7$ ($V_i$) | Sink Current Turn-on Delay | 0.5 $V_i$ to 0.9 $I_L$ (3); (4) | | 1.6 | | μs |
| $T_8$ ($V_i$) | Sink Current Rise Time | 0.1 $I_L$ to 0.9 $I_L$ (3); (4) | | 0.2 | | μs |
| fc ($V_i$) | Commutation Frequency | $I_L$ = 2A | | 25 | 40 | KHz |
| $T_1$ ($V_{en}$) | Source Current Turn-off Delay | 0.5 $V_{en}$ to 0.9 $I_L$ (2); (4) | | 3 | | μs |
| $T_2$ ($V_{en}$) | Source Current Fall Time | 0.9 $I_L$ to 0.1 $I_L$ (2); (4) | | 1 | | μs |
| $T_3$ ($V_{en}$) | Source Current Turn-on Delay | 0.5 $V_{en}$ to 0.1 $I_L$ (2); (4) | | 0.3 | | μs |
| $T_4$ ($V_{en}$) | Source Current Rise Time | 0.1 $I_L$ to 0.9 $I_L$ (2); (4) | | 0.4 | | μs |
| $T_5$ ($V_{en}$) | Sink Current Turn-off Delay | 0.5 $V_{en}$ to 0.9 $I_L$ (3); (4) | | 2.2 | | μs |
| $T_6$ ($V_{en}$) | Sink Current Fall Time | 0.9 $I_L$ to 0.1 $I_L$ (3); (4) | | 0.35 | | μs |
| $T_7$ ($V_{en}$) | Sink Current Turn-on Delay | 0.5 $V_{en}$ to 0.9 $I_L$ (3); (4) | | 0.25 | | μs |
| $T_8$ ($V_{en}$) | Sink Current Rise Time | 0.1 $I_L$ to 0.9 $I_L$ (3); (4) | | 0.1 | | μs |

1) 1)Sensing voltage can be –1 V for t ≤ 50 μsec; in steady state $V_{sens}$ min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

**Figure 1 :** Typical Saturation Voltage vs. Output Current.

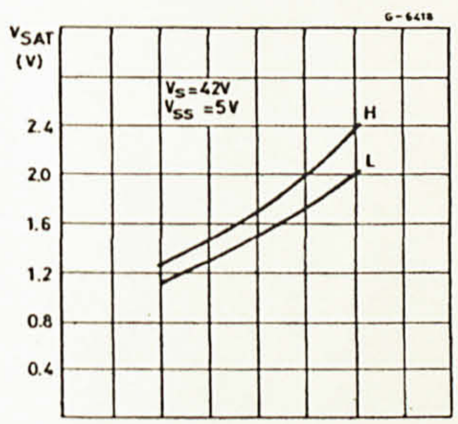
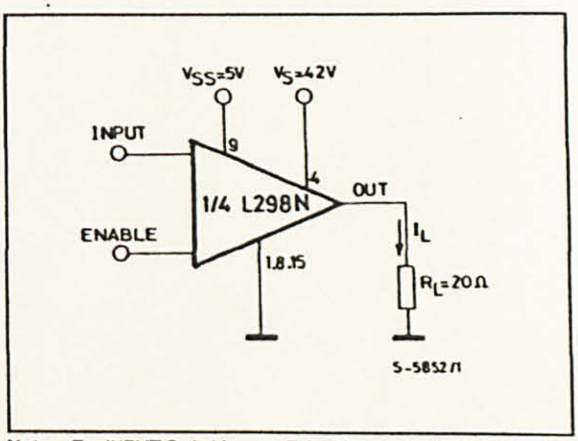



**Figure 2 :** Switching Times Test Circuits.




**Note :** For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

**Figure 3 :** Source Current Delay Times vs. Input or Enable Switching.

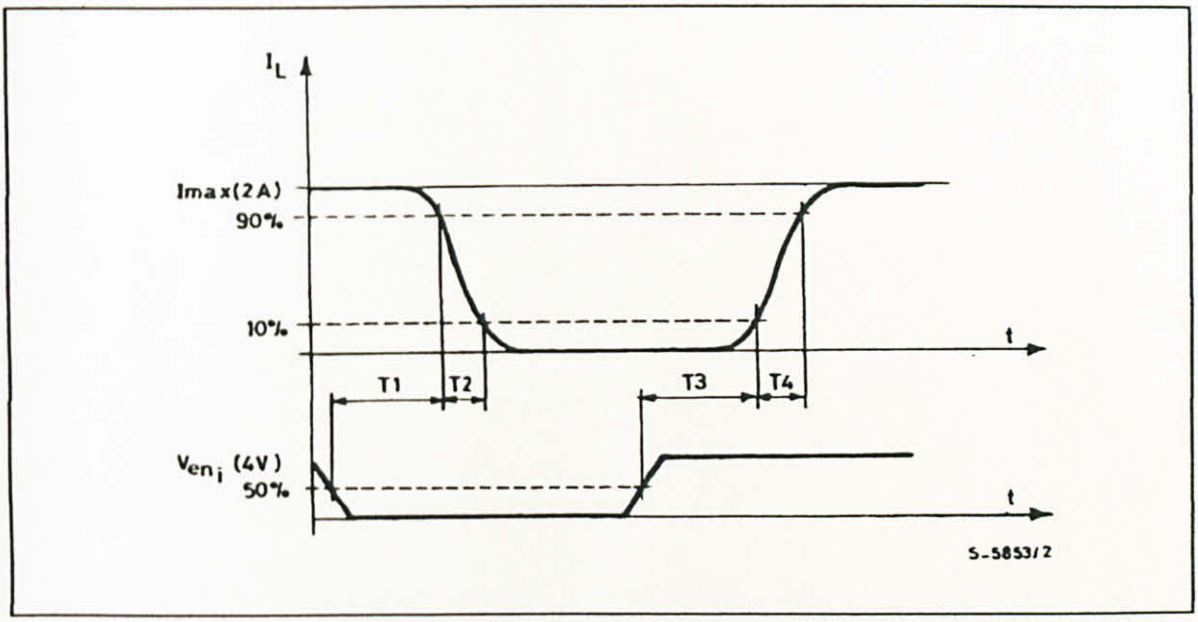
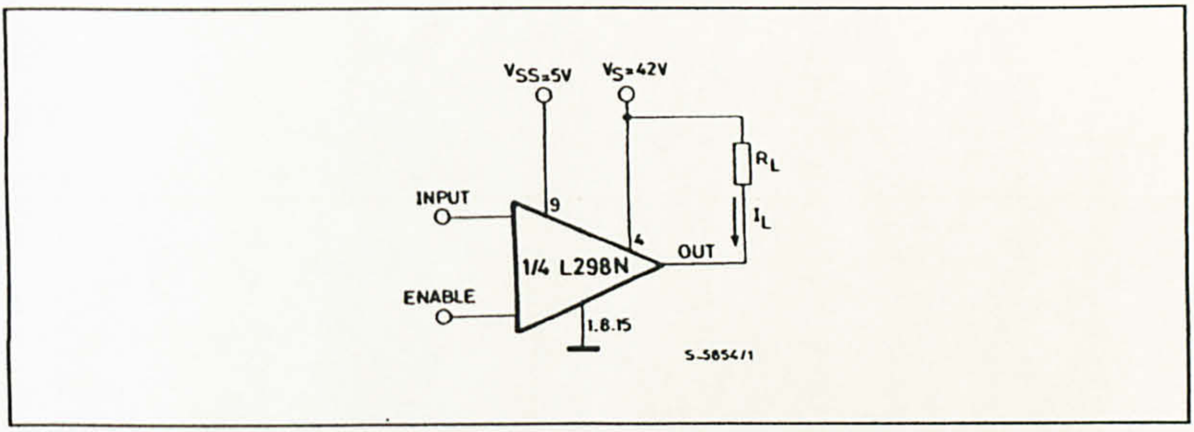



**Figure 4 :** Switching Times Test Circuits.




**Note :** For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

**Figure 5 :** Sink Current Delay Times vs. Input 0 V Enable Switching.

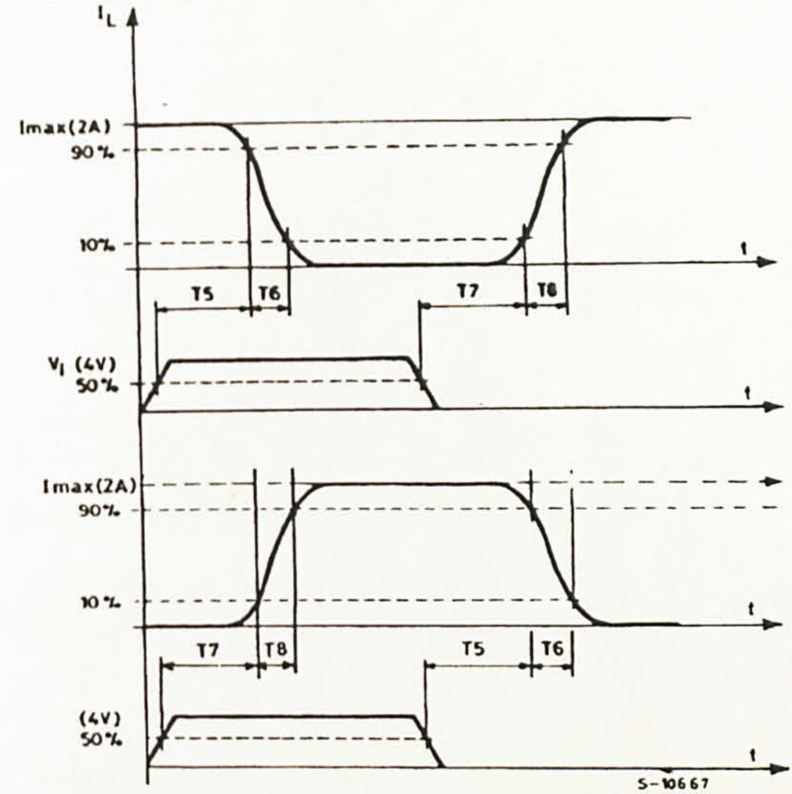
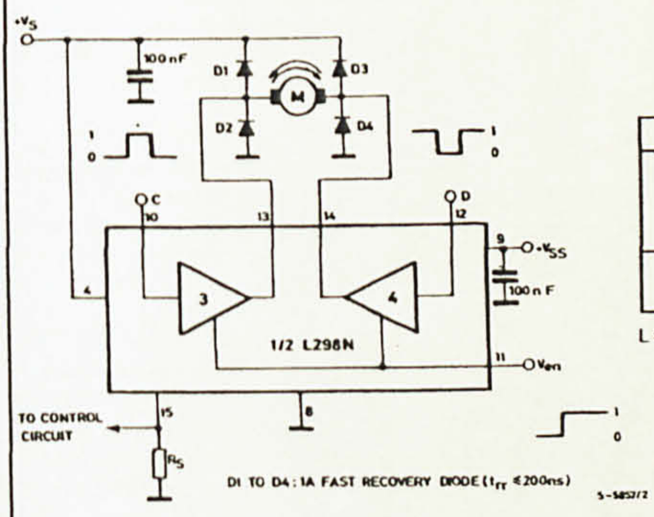


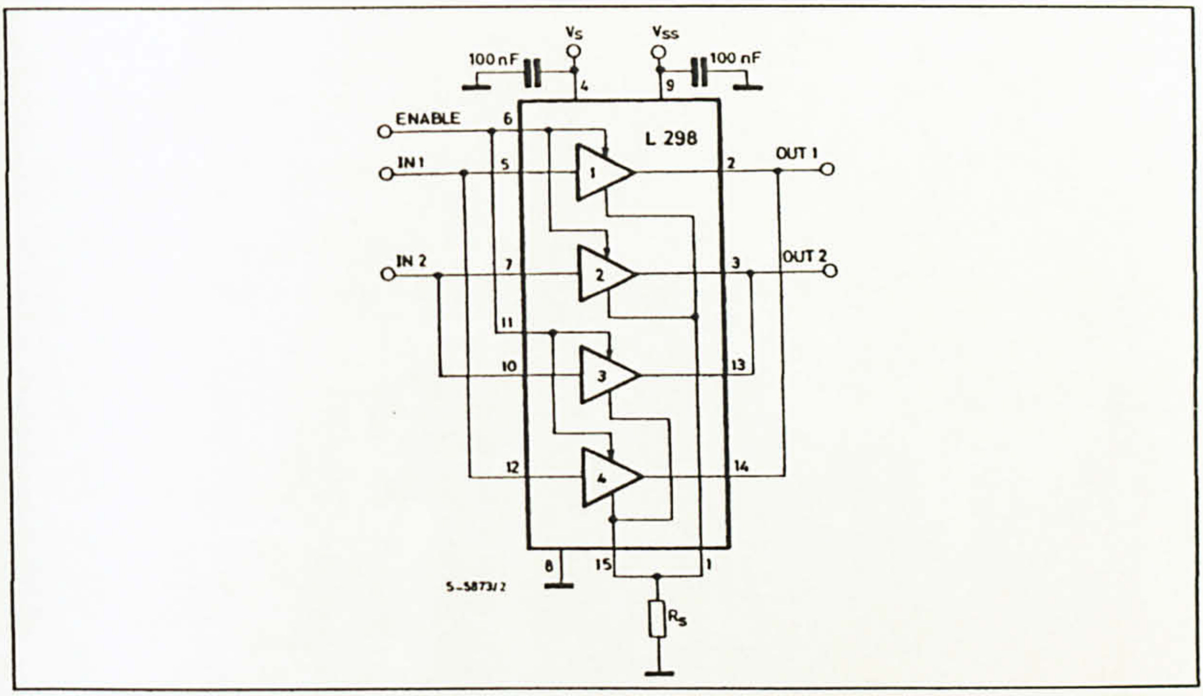

**Figure 6 :** Bidirectional DC Motor Control.




| Inputs | | Function |
|---|---|---|
| $V_{en}$ = H | C = H ; D = L | Forward |
| | C = L ; D = H | Reverse |
| | C = D | Fast Motor Stop |
| $V_{en}$ = L | C = X ; D = X | Free Running Motor Stop |

L = Low H = High X = Don't care

**Figure 7 :** For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.




## APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor ($R_{SA}$ ; $R_{SB}$.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

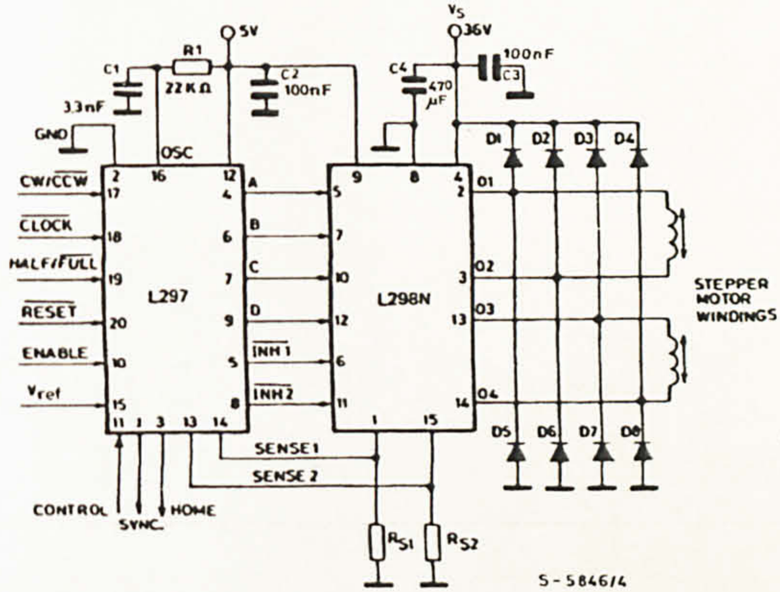
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

**Figure 8 :** Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.

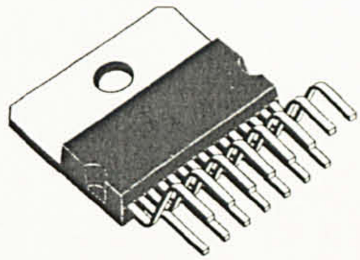



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes $\begin{cases} V_F \leq 1.2\ V @ I = 2\ A \\ trr \leq 200\ ns \end{cases}$

| DIM. | mm | | | inch | | |
|---|---|---|---|---|---|---|
| | MIN. | TYP. | MAX. | MIN. | TYP. | MAX. |
| A | | | 5 | | | 0.197 |
| B | | | 2.65 | | | 0.104 |
| C | | | 1.6 | | | 0.063 |
| D | | 1 | | | 0.039 | |
| E | 0.49 | | 0.55 | 0.019 | | 0.022 |
| F | 0.66 | | 0.75 | 0.026 | | 0.030 |
| G | 1.02 | 1.27 | 1.52 | 0.040 | 0.050 | 0.060 |
| G1 | 17.53 | 17.78 | 18.03 | 0.690 | 0.700 | 0.710 |
| H1 | 19.6 | | | 0.772 | | |
| H2 | | | 20.2 | | | 0.795 |
| L | 21.9 | 22.2 | 22.5 | 0.862 | 0.874 | 0.886 |
| L1 | 21.7 | 22.1 | 22.5 | 0.854 | 0.870 | 0.886 |
| L2 | 17.65 | | 18.1 | 0.695 | | 0.713 |
| L3 | 17.25 | 17.5 | 17.75 | 0.679 | 0.689 | 0.699 |
| L4 | 10.3 | 10.7 | 10.9 | 0.406 | 0.421 | 0.429 |
| L7 | 2.65 | | 2.9 | 0.104 | | 0.114 |
| M | 4.25 | 4.55 | 4.85 | 0.167 | 0.179 | 0.191 |
| M1 | 4.63 | 5.08 | 5.53 | 0.182 | 0.200 | 0.218 |
| S | 1.9 | | 2.6 | 0.075 | | 0.102 |
| S1 | 1.9 | | 2.6 | 0.075 | | 0.102 |
| Dia1 | 3.65 | | 3.85 | 0.144 | | 0.152 |

## OUTLINE AND MECHANICAL DATA



**Multiwatt15 V**